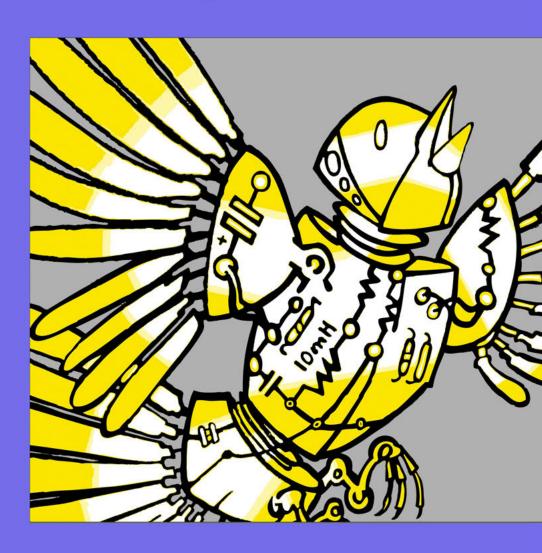
ARTIFICIAL LIFE 14



Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems

Edited by Hiroki Sayama, John Rieffel, Sebastian Risi, René Doursat, and Hod Lipson



Attribution-NonCommercial-NoDerivs 3.0 United States

You are free:





to Share — to copy, distribute and transmit the work

Under the following conditions:



Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).



Noncommercial — You may not use this work for commercial purposes.



 $\textbf{No Derivative Works} \ -- \ \text{You may not alter, transform, or build upon this work.}$

With the understanding that:

 $\begin{tabular}{ll} \textbf{Waiver} & -- \text{Any of the above conditions can be } \underline{\textbf{waived}} \text{ if you get permission from the copyright holder.} \end{tabular}$

Public Domain — Where the work or any of its elements is in the <u>public domain</u> under applicable law, that status is in no way affected by the license.

Other Rights — In no way are any of the following rights affected by the license:

- Your fair dealing or <u>fair use</u> rights, or other applicable copyright exceptions and limitations;
- The author's moral rights;
- Rights other persons may have either in the work itself or in how the work is used, such as <u>publicity</u> or privacy rights.

Notice — For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to http://creativecommons.org/licenses/by-nc-nd/3.0/us/.

This is a human-readable summary of the Legal Code (the full license) which can be found at found at http://creativecommons.org/licenses/by-nc-nd/3.0/us/legalcode.

Table of Contents

Pr	eface	ix
Ι	Keynotes	1
	Leroy Cronin: Hybrid Chemo-Robotic Systems for Embodied Chemical Evolution	
II	Oral Presentations	7
Ev	volutionary Dynamics	9
	Atsuko Mutoh, Shohei Kato and Nobuhiro Inuzuka: Evolution of Frequency-Dependent Sexual Selection Using	
	Agent-Based Model	
	Aditya Rawal, Risto Miikkulainen and Janette Boughman: Evolution of Communication in Mate Selection	
	Antoine Frenoy, François Taddei and Dusan Misevic: Constrained Genetic Architecture Promotes Cooperation	
	Kyle Harrington, Jesse Freeman and Jordan Pollack: Coevolution in Hide and Seek: Camouflage and Vision	. 25
	Carole Knibbe and David P. Parsons: What Happened to My Genes? Insights on Gene Family Dynamics from Digital Genetics Experiments	. 33
	Jeff Clune, Jean-Baptiste Mouret and Hod Lipson: Summary of "The Evolutionary Origins of Modularity"	
	Daniel Weissman: Stress-Induced Variation Can Cause Average Mutation and Recombination Rates to Be Positively	
	Correlated with Fitness	. 43
	Drew Blount: <i>Identifying Adaptations by Measuring the Trait Derivative of Fitness</i>	. 45
	A Gene Regulatory Network (GRN) Approach	. 47
	Danesh Tarapore and Jean-Baptiste Mouret: Comparing the Evolvability of Generative Encoding Schemes	
	Padmini Rajagopalan, Kay E. Holekamp and Risto Miikkulainen: <i>The Evolution of General Intelligence</i>	
	Christopher Timperley and Susan Stepney: Reflective Grammatical Evolution	
	Andrew M. Webb and Joshua Knowles: <i>Studying the Evolvability of Self-Encoding Genotype-Phenotype Maps</i> Daniel Power, Eors Szathmáry and Richard Watson: <i>Ecosystem Memory Is Emergent from Local-Level Natural</i>	. 79
	Selection	. 87
	Dominique Chu and David Barnes: Evolving Biological Systems: Evolutionary Pressure to Inefficiency	
Aı	rtificial Evolutionary Ecosystems	97
	Joachim Erdei and Borys Wrobel: Evolution of Animats Following a Moving Target in an Artificial Ecosystem	. 97
	Takashi Ito, Marcin L. Pilat, Reiji Suzuki and Takaya Arita: Population and Evolutionary Dynamics Based on Predator-Prey Relationship in 3D Physical Simulation	. 105
	Laura Grabowski and Javier Magana: Building on Simplicity: Multi-stage Evolution of Digital Organisms	
	Anya Johnson, Eli Strauss, Rodney Pickett, Christoph Adami, Ian Dworkin and Heather Goldsby: More Bang For	
	Your Buck: Quorum-Sensing Capabilities Improve the Efficacy of Suicidal Altruism	. 121
	Arthur W. Covert III, Siena McFetridge and Evan Delord: Structured Populations with Limited Resources Exhibit Higher Rates of Complex Function Evolution	. 129

Robot and Agent Behavior	135
Joshua Auerbach, Deniz Aydin, Andrea Maesani, Przemyslaw Kornatowski, Titus Cieslewski, Gregoire Heitz	
Pradeep Fernando, Ilya Loshchilov, Ludovic Daler and Dario Floreano: RoboGen: Robot Generation throug	
Artificial Evolution	
Mark Wagy and Josh Bongard: Collective Design of Robot Locomotion	
Antoine Cully and Jean-Baptiste Mouret: Learning to Walk in Every Direction with the TBR-Learning algorithm	
Jared Moore and Philip McKinley: <i>Investigating Modular Coupling of Morphology and Control with Digital Musc</i> . Sylvain Koos, Antoine Cully and Jean-Baptiste Mouret: <i>Abstract of: "Fast Damage Recovery in Robotics with the Coupling of Morphology and Control with Digital Musc</i> ."	
T-Resilience Algorithm"	
Evert Haasdijk, Nicolas Bredeche and A. E. Eiben: <i>Making MONEE</i>	158
tivist Learning System in Continuous Environments	
Matthew Egbert and Lola Cañamero: <i>Habit-Based Regulation of Essential Variables</i>	ir
Computers	
John Lones, Matthew Lewis and Lola Cañamero: Hormonal Modulation of Development and Behaviour Permits Robot to Adapt to Novel Interactions	
Nicola Capodieci, Emma Hart and Giacomo Cabri: Idiotypic Networks for Evolutionary Controllers in Virtual Create	
Jared Moore and Anthony Clark: Hold the Spot: Evolution of Generalized Station Keeping for an Aquatic Robot.	
Justin K. Pugh, Andrea Soltoggio and Kenneth O. Stanley: Real-time Hebbian Learning from Autoencoder Feature for Control Tasks	?S
Miguel Duarte, Fernando Silva, Tiago Rodrigues, Sancho Oliveira and Anders Christensen: <i>JBotEvolver: A Versatil Simulation Platform for Evolutionary Robotics</i>	le
Jorge Gomes, Pedro Mariano and Anders Christensen: Systematic Derivation of Behaviour Characterisations i	
Evolutionary Robotics	212
Soft Robotics and Morphologies	221
Nicholas Cheney, Jeff Clune and Hod Lipson: Evolved Electrophysiological Soft Robots	
Vito Cacucciolo, Matteo Cianchetti and Cecilia Laschi: A Model-Based Framework to Investigate Morphologica	
Computation in Muscular Hydrostats and to Design Soft Robotic Arms	
Mark Khazanov, Julian Jocque and John Rieffel: Evolution of Locomotion on a Physical Tensegrity Robot Michal Joachimczak, Reiji Suzuki and Takaya Arita: Fine Grained Artificial Development for Body-Controller Co	232
evolution of Soft-Bodied Animats	
Dan Lessin, Don Fussell and Risto Miikkulainen: Adapting Morphology to Multiple Tasks in Evolved Virtual Create	
Daniel Richards and Martyn Amos: Evolving Morphologies with CPPN-NEAT and a Dynamic Substrate	255
Collective Robotics	263
Lenka Pitonakova, Richard Crowder and Seth Bullock: Understanding the Role of Recruitment in Collective Robe	
Foraging	
Nicolas Bredeche: Embodied Evolutionary Robotics with Large Number of Robots	
Plamen Ivanov and Dylan Shell: Associating Nearby Robots to Their Voices	
Iñaki Fernández Pérez, Amine Boumaza and François Charpillet: Comparison of Selection Methods in On-Lin	
Distributed Evolutionary Robotics	
Becky Naylor, Mark Read, Jon Timmis and Andy Tyrrell: The Relay Chain: A Scalable Dynamic Communication	
Link between an Exploratory Underwater Shoal and a Surface Vehicle	290
Collective Behaviors	299
Eliseo Ferrante, Ali Emre Turgut, Tom Wenseleers and Cristián Huepe: Scale-Free Correlations in Collective Motio	
with Position-Based Interactions	
Olaf Witkowski and Takashi Ikegami: Asynchronous Evolution: Emergence of Signal-Based Swarming	
Patrick Haley, Randal Olson, Fred Dyer and Christoph Adami: Exploring Conditions That Select for the Evolution of Cooperative Group Foraging	
Sebastian von Mammen and Sarah Edenhofer: Swarm Grammars GD: Interactive Exploration of Swarm Dynamic	
and Structural Development	

Hiroki Sayama: Four Classes of Morphogenetic Collective Systems	. 320
Katie Bentley, Kyle Harrington and Erzsébet Ravasz Regan: Can Active Perception Generate Bistability? Heterogeneous Collective Dynamics and Vascular Patterning	. 328
Brian Hrolenok, Hanuma Teja Maddali, Michael Novitzky, Tucker Balch and Kim Wallen: Inferring Social Structure of Animal Groups From Tracking Data	
Heiko Hamann: Evolution of Collective Behaviors by Minimizing Surprise	
Andres Burgos and Daniel Polani: An Informational Study of the Evolution of Codes in Different Population Structure	
Hugues Bersini: In Between Schelling and Maynard-Smith	
Social Dynamics and Evolution	365
Arend Hintze and Masoud Mirmomeni: Evolution of Autonomous Hierarchy Formation and Maintenance	. 365
Alberto Antonioni, Seth Bullock and Marco Tomassini: REDS: An Energy-Constrained Spatial Social Network Mode Bijan Ranjbar-Sahraei, Daan Bloembergen, Haitham Bou Ammar, Karl Tuyls and Gerhard Weiss: Effects of Evolution on the Emergence of Scale Free Networks	
Eric Silverman, Jakub Bijak, Daniel Courgeau and Robert Franck: Advancing Social Simulation: Lessons from Demography	
Olaf Witkowski and Nathanael Aubert: Pseudo-Static Cooperators: Moving Isn't Always about Going Somewhere	. 392
Genki Ichinose and Hiroki Sayama: Invasion of Cooperation in Scale-Free Networks: Accumulated vs. Average Payof	
Jiin Jung and Aaron Bramson: An Agent-Based Model of Indirect Minority Influence on Social Change Miguel Gonzalez, Richard Watson, Jason Noble and Seth Bullock: The Origin of Culture: Selective conditions for	
Horizontal Information Transfer Tsubasa Azumagakito, Reiji Suzuki and Takaya Arita: Gene-Culture Coevolution of Language: Measurement-	
Interval Dependence of Evolutionary Rate	
alization of Business	
Boolean Networks, Neural Networks and Machine Learning	419
Taichi Haruna and Sayaka Tanaka: On the Relationship between Local Rewiring Rules and Stationary Out-Degree Distributions in Adaptive Random Boolean Network Models	
Octavio Zapata and Carlos Gershenson: <i>Random Fuzzy Networks</i>	. 427
inary Study	. 429
David Howard and Alberto Elfes: Evolving Spiking Networks for Turbulence-Tolerant Quadrotor Control	
Xun Li and Risto Miikkulainen: Evolving Multimodal Behavior Through Subtask and Switch Neural Networks Alessandro Fontana, Andrea Soltoggio and Borys Wrobel: POET: An Evo-Devo Method to Optimize the Weights of	. 439
Large Artificial Neural Networks	
Jean-Baptiste Mouret and Paul Tonelli: Abstract of: "On the Relationships between Generative Encodings, Regularity, and Learning Abilities when Evolving Plastic Artificial Neural Networks"	. 455
Konstantin Lakhman and Mikhail Burtsev: Evolution, Development and Learning with Predictor Neural Networks	. 457
Joshua Auerbach, Chrisantha Fernando and Dario Floreano: Online Extreme Evolutionary Learning Machines Christopher Headleand, Llyr Ap Cenydd and William Teahan: Berry Eaters: Learning Colour Concepts with Tem-	. 465
plate Based Evolution	. 473
Artificial Chemistries, Cellular Automata and Self-Organizing Systems	481
Peter Banda and Christof Teuscher: An Analog Chemical Circuit with Parallel-Accessible Delay Line for Learning	
Temporal Tasks	
Louis Kauffman and Marius Buliga: Chemlambda, Universality and Self-Multiplication	
Nathaniel Virgo, Simon McGregor and Takashi Ikegami: Self-Organising Autocatalysis	. 506
Eran Agmon, Alexander Gates, Valentin Churavy and Randall Beer: Quantifying Robustness in a Spatial Model of Metabolism-Boundary Co-Construction	
Theodore P. Pavlic, Alyssa M. Adams, Paul C. W. Davies and Sara Imari Walker: Self-Referencing Cellular Automata: A Model of the Evolution of Information Control in Biological Systems	

Nathaniel Virgo and Takashi Ikegami: There Can Be Only One: Reversible Cellular Automata and the Conservation of Genki	
Olivier Wang, Rene Doursat and Paul Bourgine: A Hybrid Off/On-Lattice Model of Emergence and Maintenance	e e
Autopoiesis	f-
Jean Disset, Sylvain Cussat-Blanc and Yves Duthen: Self-Organization of Symbiotic Multicellular Structures	
In-Vitro and In-Vivo Systems	549
Jennifer Pentz, Michael Travisano and William Ratcliff: Clonal Development Is Evolutionarily Superior to Aggregation in Wild-Collected Saccharomyces cerevisiae	549
Filippo Caschera and Vincent Noireaux: A Novel in Vitro Metabolic Scheme for the Construction of a Minima Biological Cell	555
Jakob Lykke Andersen, Christoph Flamm, Martin Hanczyc and Daniel Merkle: Towards an Optimal DNA-Template Molecular Assembler	557
Stefan Badelt, Christoph Flamm and Ivo L. Hofacker: Computational Design of a Circular RNA with Prion-Lik Behavior	
Hyunju Kim, Paul C. W. Davies and Sara Imari Walker: Informational Architecture of the Fission Yeast Cell Cycle Regulatory Network	
Evolutionary Art, Philosophy and Entertainment Tomohiro Suzuki and Yasuhiro Suzuki: Modeling and Evaluating the Process of Creating Paintings with Evolution	57]
Computing	571
Engaging Life Stories Using Virtual Worlds	580
Nuno Barreto, Luís Macedo and Licinio Roque: MultiAgent System Architecture in Orphibs II	596
Methodologies	605
David Ackley and Trent Small: Indefinitely Scalable Computing = Artificial Life Engineering	ne 614
SPARTAN	629
III Poster Presentations	639
Ritwik Biswas, David Bryson, Charles Ofria and Aaron Wagner: Causes vs Benefits in the Evolution of Prey Group Kai Olav Ellefsen: The Evolution of Learning Under Environmental Variability	649
Miguel Duarte, Sancho Oliveira and Anders Christensen: Evolution of Hierarchical Controllers for Multirobot Syst Yuri Shalygo: The Kinetic Basis of Self-Organized Pattern Formation	665
Onofrio Gigliotta and Marco Mirolli: Evolution of Communication-Based Collaborative Behavior in Homogeneou Robots	
Paul Grouchy and Gabriele M.T. D'Eleuterio: Evolving Autonomous Agent Controllers as Analytical Mathematica Models	
Marco Montalva Medel, Gonzalo Ruz and Eric Goles: Attraction Basins in a Lac Operon Model Under Differen Update Schedules	
Eugenia Schneider and Michael Mangold: Discussion of Synchronization Problems During Cell Cycle in Artificial Cell Modeling	al
Morteza Mashayekhi, Abbas Golestani, Yasaman Majdabadi Farahani and Robin Gras: An Enhanced Artificia Ecosystem: Investigating Emergence of Ecological Niches	al

Brett Calcott: Chaining Distinct Tasks Drives the Evolution of Modularity	701
Fernando Silva, Miguel Duarte, Sancho Oliveira, Luis Correia and Anders Christensen: <i>The Case for Engineering the Evolution of Robot Controllers</i>	. 703
Lance Williams: Self-Replicating Distributed Virtual Machines	711
Ryan Scott and Robin Gras: Testing the Effects of Speciation and Mutation Rates on Distance-Based Phylogenetic	
Tree Construction Accuracy Using EcoSim	
James Stovold, Simon O'Keefe and Jon Timmis: Preserving Swarm Identity Over Time	726
Danesh Tarapore, Pedro Lima, Jorge Carneiro and Anders Christensen: Optimizing the Crossregulation Model for Scalable Abnormality Detection	. 734
Chris Marriott and Jobran Chebib: The Effect of Social Learning on Individual Learning and Evolution	
Yoshihiko Kayama and Yasumasa Imamura: Self-Organizing Process and Cluster Network in the Game of Life	. 744
Wonki Lee and Daeeun Kim: Adaptive Division of Labor in Multi-Robot System with Minimum Task Switching	750
Atsushi Shibai, Saburo Tsuru, Bei-Wen Ying, Daisuke Motooka, Kazuyoshi Gotoh, Shota Nakamura and Tetsuya	
Yomo: Mutation Accumulation in Bacteria Exposed to UV Radiation	
Wonki Lee and Daeeun Kim: Task Partitioning Based on Response Threshold Model in Robot Harvesting Task	. 759
Sjriek Alers, Karl Tuyls, Bijan Ranjbar-Sahraei, Daniel Claes and Gerhard Weiss: Insect-Inspired Robot Coordina-	= -1
tion: Foraging and Coverage	. 761
Norihiro Maruyama, Atsushi Masumori, Julien Hubert, Takeshi Mita, Douglas Bakkum, Hirokazu Takahashi and Takashi Ikegami: <i>Designing a Robotic Platform Controlled by Cultured Neural Cells</i>	. 769
Sergey Muratov, Konstantin Lakhman and Mikhail Burtsev: Neuroevolution of Sequential Behavior in Multi-Goal Navigation Task	. 771
Yuki Takeichi, Kazutoshi Sasahara, Reiji Suzuki and Takaya Arita: Twitter as Social Sensor: Dynamics and Structure	
in Major Sporting Events	. 778
Miguel Duarte, Sancho Oliveira and Anders Christensen: Hybrid Control for Large Swarms of Aquatic Drones	. 785
L. B. Soros and Kenneth O. Stanley: <i>Identifying Necessary Conditions for Open-Ended Evolution through the Artificial Life World of Chromaria</i>	. 793
Tarek Ababsa, Noureddine Djedi, Yves Duthen and Sylvain Cussat-Blanc: Splittable Metamorphic Carrier Robots.	801
Shota Onoda, Shohei Kato and Atsuko Mutoh: The Effect of the Network Structure Differences on the Diffusion of Items.	
Barak Shenhav: An Artificial Chemistry for the 'Lipid World' Scenario	
Micah Brodsky: Spatial Patterning with the Rule of Normal Neighbors	
Nawwaf Kharma and William Buckley: CodeRouge: A Project to Evolve Life-Like Autonomous Programs	
Jessica Lowell and Jordan Pollack: The Effect of Connection Cost on Modularity in Evolved Neural Networks	
Jessica Lowell, Kyle Harrington and Jordan Pollack: The Resilience of a Swarm Ecosystem Under Environmental	
Variation	827
Joshua Hecker and Melanie Moses: Real-Time Evolution of iAnt Robot Foraging Strategies	835
Christoph Salge and Daniel Polani: Don't Believe Everything You Hear: Preserving Relevant Information by Discarding Social Information	. 837
Morgan Mclaughlin and Mark Wineberg: The Effect of Network Structure on the Spatial Coevolutionary GA	845
Andrew Nelson and Brenae Bailey: Evolution as a Random Walk on a High-Dimensional Manifold Defined by Physical Law: Implications for Open-Ended Artificial Life	. 847
Haifa Rabai, Rodolphe Charrier and Cyrille Bertelle: Close Returns Plots for Detecting a Chaotic Source in an	
Interaction Network	849
William Hurndall, Richard Watson and Markus Brede: Pre-Template Metabolic Replicators: Genotype-Phenotype	
Decoupling as a Route to Evolvability	856
Antoine Hiolle, Matthew Lewis and Lola Cañamero: A Robot That Uses Arousal to Detect Learning Challenges and Seek Help	864
Alan Winfield: Estimating the Energy Cost of (Artificial) Evolution	
Matthew Francisco, Ian Wood, Selma Sabanovic and Luis Rocha: Designing a Minimalist Socially Aware Robotic	3,2
Agent for the Home	876
Adam Wang, Jacob Gold and Kyle Harrington: Feedback Control of Evolving Swarms	
Ioan Muntean: Computation and Scientific Discovery? A Bio-Inspired Approach	
Paul Szerlip and Kenneth O. Stanley: Steps Toward a Modular Library for Turning Any Evolutionary Domain into	
an Online Interactive Platform	900

Αι	uthor Index	989
	tromagnet	987
	Atsushi Masumori, Masayoshi Mitsui and Hiroya Tanaka: Designing a Passive Folding String Device with an Elec-	
	Plants	983
	Megumi Sakai and Yasuhiro Suzuki: Evolution of Chemical Signals in Ecological System Evoked by the "Cry-Wolf"	
	Mads Buch and Steen Rasmussen: Blueprint for Self-Replicating and Self-Assembling 3D Printers	981
	User Preferences	973
	Anton Bernatskiy, Gregory S. Hornby and Josh Bongard: Improving Robot Behavior Optimization by Combining	
	Networks for Temporal Pattern Recognition in the Presence of Noise	965
	Ahmed Abdelmotaleb, Neil Davey, Maria Schilstra, Volker Steuber and Borys Wrobel: Evolving Spiking Neural	, , ,
	Yutetsu Kuruma, Hideaki Matsubayashi and Takuya Ueda: <i>In Vitro Reconstruction of Functional Membrane</i>	
	MapReduce	957
	Sergi Canyameres Masip and Doina Logofatu: Framework for Adaptive Swarm Simulation and Optimization Using	777
	Interaction Closure	949
	Martin Biehl, Christoph Salge and Daniel Polani: <i>Towards Designing Artificial Universes for Artificial Agents Under</i>	941
	Insook Choi and Robin Bargar: Sounds Shadowing Agents Generating Audible Features from Emergent Behaviors	
	Marco A. Montes de Oca and Louis Rossi: A Continuous-Time Binary Consensus Protocol with Hysteretic Units Applied to the Density Classification Problem	024
	Matthew Lewis, Antoine Hiolle and Lola Cañamero: Pleasure, Persistence and Opportunism in Action Selection	932
	Morgan Mclaughlin and Mark Wineberg: The Effects of Elitism on Spatial Coevolutionary GAs	
	Jason Yoder and Larry Yaeger: Evaluating Topological Models of Neuromodulation in Polyworld	
	Brent Eskridge and Ingo Schlupp: Effects of Personality Distribution on Collective Behavior	

Preface

This volume is the proceedings of ALIFE 14: The Fourteenth International Conference on the Synthesis and Simulation of Living Systems, held July 30th to August 2nd, 2014, in Manhattan, New York (http://alife14.org/, or http://alife2014.alife.org/). ALIFE is the largest conference series focusing on the study of Artificial Life. Since its inception led by Chris Langton in 1987, it has been held at various international locations every other year, alternating with ECAL (European Conference on Artificial Life) since 1991. The history of those conferences and past websites can be found on the website of the International Society for Artificial Life (ISAL, http://alife.org/).

ALIFE 14 attracted a total of 203 submissions. Following the past ALIFE and ECAL rules, we solicited submissions in two formats: full paper (up to 8 pages) and extended abstract (2 pages). The extended abstract format was implemented in 2008 (ALIFE 11) to promote participation of researchers working in disciplines where conference submissions require abstracts only. This format may be used for presenting either original work or recently published work. Both full papers and extended abstracts were peer-reviewed by multiple (typically three to four) Program Committee members. As a result, we accepted 102 submissions for oral presentations and 62 submissions for poster presentations (oral acceptance rate: 50.2%).

The program of ALIFE 14 is composed of a number of scientific/professional/creative activities, including:

- Five keynote talks given by the following internationally renowned speakers:
 - Naomi Leonard (Princeton University, USA)
 - Jesse Louis-Rosenberg (Nervous System, USA)
 - Karl Sims (Digital Media Artist, USA)
 - Leroy (Lee) Cronin (University of Glasgow, UK)
 - John H. Conway (Princeton University, USA)
- Twenty parallel sessions on the following topics:
 - Evolutionary dynamics (3 sessions)
 - Robot and agent behavior (3 sessions)
 - Boolean networks, neural networks and machine learning (2 sessions)
 - Artificial chemistries, cellular automata and self-organizing systems (2 sessions)
 - Collective behaviors (2 sessions)
 - Social dynamics and evolution (2 sessions)
 - In-vitro and in-vivo systems
 - Artificial evolutionary ecosystems
 - Soft robotics and morphologies
 - Collective robotics
 - Evolutionary art, philosophy and entertainment
 - Methodologies
- A poster session with a plenary "Poster Blitz Movie Show"

• Nine workshops:

- Computing with Biomolecules: From Network Motifs to Complex and Adaptive Systems (Organizers: Christof Teuscher, Darko Stefanovic, Milan N. Stojanovic)
- CoSMoS 2014: 7th Workshop on Complex Systems Modelling and Simulation (Organizers: Susan Stepney, Paul Andrews)
- Exploiting Synergies Between Biology and Artificial Life Technologies: Tools, Possibilities, and Examples (Organizers: Kasper Stoy, Martin Hanczyc, Ioannis Ieropoulous, Michal Wagner, Leroy Cronin)
- Human-Machine Affective Coordination (Organizers: Luisa Damiano, Paul Dumouchel, Hagen Lehmann)
- Information Theoretic Incentives for Artificial Life (Organizers: Christoph Salge, Keyan Ghazi-Zahedi, Georg Martius, Daniel Polani)
- MEW 2014: Fourth Morphogenetic Engineering Workshop (Organizers: René Doursat, Hiroki Sayama)
- SB-AI 2014: What Can Synthetic Biology Offer to (Embodied) Artificial Intelligence?
 (Organizers: Luisa Damiano, Yutetsu Kuruma, Pasquale Stano)
- WebAL1: Artificial Life and the Web
 (Organizers: Tim Taylor, Josh Auerbach, Josh Bongard, Jeff Clune, Simon Hickinbotham, Greg Hornby)
- Third Workshop on the Evolution of Physical Systems (Organizers: John Rieffel, Jean-Baptiste Mouret, Nicolas Bredeche, Evert Haasdijk)

• Six tutorials:

- Algorithmic Graph Chemistry in Action (Instructors: Jakob L. Andersen, Christoph Flamm, Daniel Merkle)
- An Introduction to Statistical Analysis of Alife Experiments (Instructor: Mark Wineberg)
- Creating High-Quality Plots with Python and Matplotlib (Instructor: Jean-Baptiste Mouret)
- Evolutionary Robotics
 (Instructors: Nicolas Bredeche, Stéphane Doncieux, Jean-Baptiste Mouret)
- Evolving Neural Networks (Instructor: Risto Miikkulainen)
- In Silico Experimental Evolution with the Aevol Software (Instructors: Guillaume Beslon, Carole Knibbe, David P. Parsons, Dusan Misevic)
- The first ISAL summer school:
 - Historical and Philosophical Perspectives on Artificial Life (instructor: Mark Bedau)
 - Unconventional Computing (instructor: Susan Stepney)
 - Digital Evolution (instructor: Charles Ofria)
 - Evolutionary Developmental Robotics (instructor: Josh Bongard)
- A science visualization competition
- A career advising luncheon for postdocs and graduate students
 - ... and other social events

Needless to say, organizing a conference requires a lot of work, and we would not have been able to achieve our goal without help from a number of individuals and organizations. First and foremost, our highest gratitude goes to Craig Ryan, our event coordinator, for his heroic work in essentially all aspects of the conference, including negotiations with the conference venue, handling registrations, and making millions of other logistic arrangements. We are also very fortunate to have had Emily

Ryan design the impressive "ALIFE 14 bird" artwork used on the conference website and the cover of this proceedings volume, which has become a unique, memorable visual identity to the conference. We also thank Ira Fraitag and Mike Blain for their help.

To maintain a high standard of scientific quality for the conference, it is critical to have a rigorous peer review process to evaluate submissions. Luckily, we were able to rely on a very cooperative, hard working Program Committee and other reviewers who have provided constructive comments on the submissions in a timely manner (individual names listed below in an alphabetical order).

Program Committee of ALIFE 14:

Angelo Cangelosi Hussein Abbass David Ackley Peter Cariani Andy Adamatzky Timoteo Carletti Christoph Adami Pedro Castillo Andreas Albrecht José M. Cecilia Bradly Alicea Marin Cenek Fernando Almeida e Costa Gregory Chirikjian Lee Altenberg Sung-Bae Cho Martyn Amos Insook Choi Timothy Andersen Dominique Chu Claes Andersson Brian D. Connelly Takaya Arita David Cornforth Joshua E Auerbach Luis Correia Wolfgang Banzhaf Ernesto Costa Thomas Barbalet Art Covert Jeffrey Barrick Leroy (Lee) Cronin Jacob Beal Sylvain Cussat-Blanc Mark Bedau Thomas Dandekar Randall Beer Christian Darabos Roman Belavkin Kerstin Dautenhahn Mariana Benitez James Decraene Katie Bentley Jordi Delgado Peter Bentley Ralf Der Hugues Bersini Gianni Di Caro Guillaume Beslon Peter Dittrich Navneet Bhalla Stéphane Doncieux Erik Billing Alan Dorin Eleonora Bilotta Alastair Droop **Daniel Bisig** Boris Duran Tim Blackwell Yves Duthen Alan Blair James Dyke Christian Blum Matthew Egbert Margaret Eppstein Johan Bollen Josh Bongard Brent Eskridge Terry Bossomaier Harold Fellermann Amine Boumaza Christoph Flamm Paul Bourgine David Fogel Markus Brede Gary Fogel Nicolas Bredeche Robert Freitas John Bryden Tom Froese David Bryson Rudolf M. Füchslin Thomas Buhrmann Simon Garnier Larry Bull Nicholas Geard Seth Bullock Philip Gerlee

Lola Cañamero

Mario Giacobini Sherri Goings Heather Goldsby Jonatan Gomez Ángel Goñi Moreno Erik Goodman Charles Goodnight Jerzy Gorecki Laura Grabowski Robin Gras Hugo Gravato Marques Patrick Grim Thilo Gross Pauline C Haddow Heiko Hamann Martin M. Hanczyc Emma Hart Taichi Haruna Inman Harvey Robert Heckendorn J. Michael Herrmann Simon Hickinbotham Arend Hintze Thomas Hinze Paulien Hogeweg Gregory Hornby Kazufumi Hosoda Cristian Huepe Paul Humphreys Phil Husbands Tim Hutton Genki Ichinose Fumiya Iida Hiroyuki Iizuka Takashi Ikegami Yasumasa Imamura Eduardo Izquierdo Christian Jacob Klaus Jaffe Yaochu Jin Michal Joachimczak Sardanyés Josep George Kampis Yoshihiko Kayama Jozef Kelemen

David Knoester Marcelo Kuperman Yutetsu Kuruma Renaud Lambiotte Joel Lehman Tom Lenaerts Lukas Lichtensteiger Soo Ling Lim Xiaoxia Nina Lin Kristian Lindgren Joseph Lizier Daniel Lobo Robert Lowe Herve Luga Bruce MacLennan George Magoulas Narine Manukyan Georg Martius Joanna Masel Jerzy Maselko John Mccaskill Bob McKay Philip Mckinley Barry McMullin Peter McOwan Juan J. Merelo Daniel Merkle Rob Mills Jeremy Minty Dusan Misevic Shuhei Miyashita Pierre-Alain Monnard Sara Montagna Jean-Marc Montanier Melanie Moses Jean-Baptiste Mouret Andres Moya Joshua Nahum Chrystopher Nehaniv Peter Nielsen Martin Nilsson Jacobi Stefano Nolfi Shin-Ichiro Nomura Surya Nurzaman

Carole Knibbe

Carlos Gershenson

Oliver Obst William Ratcliff Kasper Stoy Tatsuo Unemi Charles Ofria Thomas Ray John Sullins Edgar Vallejo Yuta Ogai Craig Revnolds Hideaki Suzuki Sergi Valverde Mizuki Oka Danny Richards Andrew Vardy Keisuke Suzuki **Eckehard Olbrich** Luis M. Rocha Reiji Suzuki Jeffrey Ventrella Randal Olson Andrea Roli Yasuhiro Suzuki David Vernon Bjørn Østman **Eors Szathmary** Nathaniel Virgo Adam Rotaru Sebastian von Mammen Elizabeth Ostrowski Kepa Ruiz-Mirazo Nobuto Takeuchi Pierre-Yves Oudever Aaron Wagner **Erol Sahin** Uwe Tangen Andreas Pape Christoph Salge Charles Taylor Richard Watson Lael Parrott Francisco C. Santos Tim Taylor Justin Werfel Kazutoshi Sasahara Jason Teo Uri Wilensky Jonathan Pascalie Janet Wiles Joshua Payne Thomas Schmickl Christof Teuscher Dov Pechenick Roberto Serra Guy Theraulaz Lance Williams Alexandra Penn Jae Kyun Shin Serge Thill Paul Williams Robert Pennock Eric Silverman Jon Timmis Borys Wrobel Michael Travisano Ferdinand Peper Lee Spector Andrew Wuensche Russell Standish Simone Pigolotti Vito Trianni Lidia Yamamoto Marcin Pilat Kenneth Stanley Soichiro Tsuda Xin-She Yang Tetsuva Yomo Daniel Polani Pasquale Stano Elio Tuci Simon Powers Kathleen Steinhofel Andy Tyrrell Luis Zaman Mikhail Prokopenko Susan Stepney Jon Umerez Hector Zenil

Other Reviewers:

Anton Bernatskiy Nazim Fates Trevor Hinkley Stéphane Sanchez Moritz Buck Matthew Fricke Neal Holtshulte Danesh Tarapore Andrés Bueno The Anh Han Nikhil Joshi Mark Wagy

Alexandre Campo Joshua Hecker Koichi Nishiyama

We are very grateful to the following sponsors for their financial and in-kind support for the conference, which was essential in making ALIFE 14 a successful event:

- US National Science Foundation
- MIT Press
- Cornell University
- Thomas J. Watson School of Engineering and Applied Science at Binghamton University
- Wolfram Research

Finally, we would like to express our sincere thanks to keynote speakers and contributing authors whose quality presentations are what make ALIFE 14 a truly unique, outstanding academic meeting.

We hope you will enjoy the breadth and depth of various topics covered in this volume. They collectively represent the state of the art of Artificial Life, a vibrant intellectual melting pot stirred and served fresh in the middle of Manhattan.

June 2014

Hiroki Sayama John Rieffel Sebastian Risi René Doursat Hod Lipson

Part I Keynotes

Hybrid Chemo-Robotic Systems for Embodied Chemical Evolution

Leroy Cronin¹

¹Complex Chemical Systems Lab, School of Chemistry, University of Glasgow, Glasgow, G12 8QQ, UK.

Lee.cronin@Glasgow.ac.uk

Abstract

The development of inorganic systems capable of evolution as a fundamentally less complex 'emergent' model of prebiotic evolution is proposed as a solution to the information paradox that could pre date other more complex systems e.g. heteropolymers capable of catalysis and replication. In this approach we allow complexity and information transfer between systems to bootstrap the assembly of systems capable of exhibiting evolutionary dynamics. By developing a new paradigm for evolution outside of biology, we propose a roadmap to engineer the emergence of the minimal evolvable inorganic chemical entities that could eventually make the transition from dead to living matter. This new experimental approach not only requires developments in information theory, but automation of experiments, robotics, and evolutionary programming. Thus this area defines an intrinsically multidisciplinary field requiring contributions from chemistry / chemical engineering, molecular biology / genetics, robotics and computing science. Success could mean new hints to the origin of life on earth as well as the possibility of developing new types of artificial life based upon chemistries not found in biology today.

Introduction

What is life? How did life start on planet earth and how hard was the process to get going? Also, can we engineer totally new types of life forms based upon a different 'chemistry set'? These are fantastic questions that continue to capture the imaginations of many researchers but we wanted to go further and ask if it is possible to create life in the laboratory from scratch on a reasonable time scale. Indeed how would an experimentalist go about it and in what form?

To answer this, in our laboratory we are currently developing a new conceptual approach to engineer evolution outside of biology, inspired by recent work (Joyce 2012; Kauffman 2011; Deamer et al. 2011). We will then use this to develop an experimental framework to test the evolution first hypothesis. We anticipated that this research will lead to the search for minimal evolvable inorganic chemical entities using a series of experimental platforms. This is because minimal self-assembling inorganic systems capable of catalysis and replication may provide a route to cross the information threshold where the number of evolvable bits (Evb) exceeds that required to start the process (Ib). Ultimately this approach could allow us to (re)discover biology relevant to life on earth

as it is today, or to develop a totally new 'inorganic biology' (Cronin 2013). Similar to Joyce (Joyce 2012), we think that the evolvable pre-biotic inorganic systems could be considered to be minimal life forms where Evb>Ib and could therefore represent a testable approach to explore plausible inorganic 'origins' relevant to the emergence of life on Earth, as well as allowing the engineering of new biologies from the bottom up (expanding the possible interplanetary chemistries capable of life). In this context we consider life to be a replicative population-based ensemble of unstable entities, capable through evolution of giving a fitter population as a function of generations capable of survival. Such entities, through generational survival, naturally acquire information content with a measureable increase in functional 'bit-content' (at least initially). Our intention is that this will provide a roadmap to progress towards the complexity of contemporary biological systems whereby the key step is the transition to evolvability, and then ultimately to open ended evolution, see Figure 1.

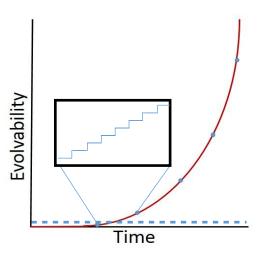


Figure 1. Graph depicting the emergence of evolution with graph of evolvability against time. One of the big questions is the nature of any event / transition in terms of the individual steps needed to be taken to lead to evolutionary chemical systems.

The Roadmap

Although we have many possible starting points for this project, the issues about timescale, evidence of success, a testable theory, and architecture of the chemical search engine are all currently rather challenging. In our work we have been inspired to develop a roadmap that will lead to the establishment of this endeavor in our laboratory over the next years, including:

- 1) A theoretical description of the minimal evolving inorganic system produced by making the 'transition to evolution'. This will define the boundary conditions on the number of operations, information content of the chemical system, and observable behaviour predicted enroute to autonomously evolving chemical systems.
- 2) A chemical map, best described as a chemical network allowing nodes of interacting molecules to be transformed by chemical reactions (connecting the nodes), in inorganic space, along with a language to define and interrogate the map.
- 3) Engineering of hardware allowing the chemical network to be explored. This requires modular plug and play reactor units with well-defined inputs (chemicals, physical control, on board logic and analytical sensors), well defined outputs, and operations resulting from the network analysis.
- 4) Construction of the massively parallel chemical array integrating the network language from (2), the hardware from (3), and the use of theory developed in (1). Over the chemical array, a series of chemical search algorithms will implemented in the search for evolvability.

Ultimately we aim to engineer a minimal chemical cell that can undergo evolution by the process of metabolism, growth, replication, and selection of the fitness through, and feedback of, trillions upon trillions of interactions. The aim will be to 'evolve' through the universe of chemistry, and discover a minimal chemical entity that can emerge as the first candidate for a truly artificial / minimal life form.

Abiotic Evolutionary Chemical Engine

To start to address this grand roadmap in our laboratory we have recently developed a programmable fluidic system comprising a series of linear flow systems, coupled network reactor arrays with a sensor array, and control actuation (Cronin 2012). In this research, by coupling separate reactions in both space and time, we aim to control the assembly of kinetically unstable structures that are maintained away from thermodynamic equilibrium. In a nutshell we will use a fully automated semi-batch-semi-flow system to explore the non-equilibrium assembly of inorganic structures, combinatorial organic reactions combined with gradients of light, pH, redox, organic cations selecting for

catalysis, emergence functionality and persistence of morphology, see Figure 2.

Thus, by generating a vast population of chemical inputs, stochastic chemical reactions, followed by an assay that will allow selection and feedback, the aim will be to develop a new paradigm for evolution outside of biology. The system will use evolutionary programming algorithms to search the chemical space. The aim is to utilize an evolutionary process to boot-strap the discovery of minimal inorganic chemical cells (ichells) capable of evolutionary dynamics. This will require new developments in high throughput chemistry, screening, computational technology, and theoretical developments to adequately set the test question. In effect, this chemical search engine will aim to cover, using advanced algorithms, a chemical space similar to that which was available on planet earth before and during the onset of life, as well as exploring other plausible chemical universes for alternative and new biologies based upon an entirely new and unexpected chemistry set. Our hypothesis is that, by engineering the minimal chemical entity that can undergo evolution autonomously in a given environment, we will provide a new and exciting route to understanding the origin of life as well as new artificial life forms with novel chemistries.

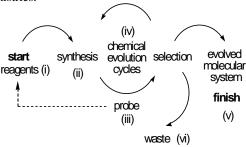


Figure 2. Chemical Evolution Reagents (i) are added to custom reactors (ii) and the results are measured with a probe of sensor arrays (iii). Evolutionary algorithms dictate whether to continue reaction cycles (with or without additional inputs), to collect the material (v), or to eject it to waste (vi). Multiple such modules are put in series and parallel for complex explorations and metabolism engineering will be done using flow control, as well as embedded redox active reagents setting up electrical and proton gradients.

By focusing on the transition region between dead and living matter it is hoped that this vision will attract an unprecedented range of collaborating scientists all unified by the grand vision of creating life in the lab.

Acknowledgments. I'd like to acknowledge my research group, past and present (see www.croninlab.com) as well as our collaborators, colleagues, and many funders in particular the EPSRC whose funding has made this research possible.

References

Joyce G F (2012) Bit by bit: The Darwinian Basis of Life Plos Biology,

Kauffmann S (2011) Approaches to the origin of life on earth. Life 1:3448

- Olasagasti F, Kim HJ, Pourmand N, Deamer DW (2011) Non-enzymatic transfer of sequence information under plausible prebiotic conditions, Biochimie. 93:556-561
- Cronin L (2013) Linking Evolution in Silico, Hardware, and Chemistry to discover or engineer Inorganic Biology. Advances in Artificial Life ECAL 12:1066-1067Robinson, A. L. (1980a). New ways to make microscopic and the state of the state o
- Cronin L et al (2012) A flow-system array for the discovery and scale up of inorganic clusters. Nature Chemistry, 4: 1037-1043

Part II Oral Presentations

Evolutionary Dynamics

Evolution of Frequency-Dependent Sexual Selection Using Agent-Based Model

Atsuko Mutoh, Shohei Kato and Nobuhiro Inuzuka

Nagoya Institute of Technology, Japan atsuko@nitech.ac.jp

Abstract

Nonindependent mate choice occurs when a female is influenced in her choices of mate by the social environment. Frequency-dependent selection (FDS) is a typical example of a nonindependent mate choice and comes in two forms: positive or negative. In the positive form, any rare variant is at a disadvantage, whereas rare variants are favored in the negative form. Both forms of FDS have been confirmed in many species, and several mathematical and theoretical biology studies have reported the advantages of each. However, few studies have focused on the evolution of the two forms of FDS together. In this work, we simulated FDS using an agent-based model consisting of imported mating strategy as gene and female preference influenced by the social environment as meme. Experimental results revealed a relation between the operational sex ratio of males and the FDS strategy of females. A similar tendency was observed among real an-

Introduction

Most models of sexual selection assume that females exhibit a genetically based preference for males with specific traits (Kirkpatrick and Ryan, 1991). However, female mate choice is a complex process involving not only genetic but also nongenetic factors (Westneat et al., 2000). There is increasing evidence that social factors have an important influence on mate-choice decisions (Witte and Noltemeier, 2002). Numerous studies have reported that social environment may influence mate preferences.

Individuals usually mate with opposite-sex others based on their own assessment of the suitability prospective mate, but this assessment can also be modulated by observing the decisions of others, so-called nonindependent mate choice (Vakirtzis and Roberts, 2010). One example of nonindependent mate choice is frequency-dependent selection (FDS). Positive FDS is a form of selection in which genotypes are favored when they are common and negative FDS is a form of selection in which genotypes are favored when they are uncommon (Hughes et al., 2013). Many experimental studies have confirmed positive FDS and negative FDS in nature separately (Singh and Sisodia, 2000; Vakirtzis, 2011).

There are also many studies on FDS using agent-based models in the fields of artificial life. Kawata et al. have examined the possibility of speciation without viability selection by positive FDS (Kawata and Yoshimura, 2000), and Kokko et al. have investigated how negative FDS evolved (Kokko et al., 2007). Some studies have demonstrated the advantages of mate-choice copying (Sirot, 2001; Dubois, 2007). However, these studies are tightly focused and there is no research on both positive and negative FDS for mate choice, together.

Guppies are known to copy the mate choice of other females (Dugatkin, 1992), and has been said that such matechoice copying produces positive FDS (Kirkpatrick and Dugatkin, 1994). However, there have been cases in which guppies have exhibited negative FDS behavior (Hughes et al., 2013). This indicates that different forms of FDS occur in the same species. We expect that the FDS strategy is dictated by the individuality of the agents. Servedio and Kirkpatrick have shown that the mate-choice copying is hereditary (Servedio and Kirkpatrick, 1996), and Dubois defined the copier female and non-copier female to choose a mate by nature (Dubois, 2007). In this work, we propose an agent-based model for both positive and negative FDS to study how to evolute.

Until now, we have expressed inborn bodily characteristics as genes and acquired preferences as memes. In the previous work, we configured an evolutionary model of artificial life (agents) that combine genes and memes and observed their influence on changes in preferences concerning mate selection (Tokuhara et al., 2005; Mutoh et al., 2010). In this paper, we propose a model that adds genes expressing the mating strategy of the female in order to determine how social factors influence female mate choice. By doing so, we can computationally observe mate selection behavior by agents and discuss the evolution of mating strategy on the basis of agent responses to the environment as generations proceed. Finally, we compare our proposed model with 32 kinds of real animals.

Agent Model

We have previously described an enhanced Lerena model (Lerena, 2000) in the form of an agent-based model consisting of both hereditary traits (genes) and acquired traits (memes) (Tokuhara et al., 2005; Mutoh et al., 2010). The concept of memes was proposed by R. Dawkins (Dawkins, 1989), who defined a meme as both a base factor and a unit of cultural information. Our agent model was able to represent constant (i.e., hereditary) and variable (i.e., acquired) information as genes and memes, respectively. However, previous models had no individual difference to change their memes. In this paper, we describe a new model that reflects the concept of FDS and how to change memes is decided by their genes.

Agents

An agent a_i consists of sex sex_i , age aqe_i , dyad genes $qene_i$, and meme $meme_i$, as

$$a_i(sex_i, age_i, gene_i, meme_i).$$
 (1)

Genes are hereditary. The first one is for inborn gene traits and the second for newly added ones related to mating strategy. Memes relate to preferences for gene traits.

$$gene_i = (g_i^{trait}, g_i^{strategy}),$$
 (2)
 $meme_i = meme_i^{pref},$ (3)

$$meme_i = meme_i^{pref},$$
 (3)

where g_i^{trait} is a gene trait of an agent a_i , $g_i^{strategy}$ is a mating-strategy gene of an agent a_i , and $meme_i^{pref}$ is the preference of the q^{trait} in mate choice. The expression of the mating-strategy gene and meme preference is limited to females ($sex_i = female$) and the expression of the gene trait is limited to males ($sex_i = male$). For the sake of simplicity, we set the trait of each male to be fixed throughout his life; that is, males do not have a meme trait.

Mating-strategy genes

We think that common males likely to be popular males. This is because it is easy for popular males to leave offspring. We assume that whether a female prefers popular or unpopular males is decided by her hereditary gene; that is, her mating-strategy gene.

We assume a gene $g^{strategy}$ encoded by bit-strings that determine the mating-strategy of females but is not expressed in males. The length of strings is one bit (0 or 1). When the $g^{strategy}$ of a female agent is one, her matingstrategy is positive FDS, which means she prefer males with popular traits. Conversely, when the $g^{strategy}$ of a female agent is zero, her mating-strategy is negative FDS, which means she prefers males with unpopular traits. We describe later how to discriminate between popular and unpopular

As mentioned above, we represent individuals having traits and mating-strategies as genes and preferences as memes.

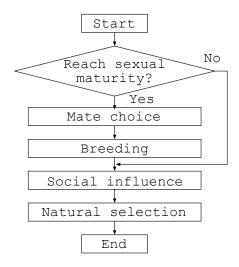


Figure 1: A flow of agent actions during each step.

Action

A single run for each step is mostly a repetition of the following three procedures:

- 1. Mate choice
- 2. Breeding
- 3. Social influence

A flow of agent actions is shown in Figure 1. First, each female reaching sexual maturity selects a male mate on the basis of her preference. Males and females whose mates are determined by mate choice then breed. After breeding, the mate preference of each female is influenced by social environment. This means that the meme preference of each female is rewritten. Finally, agents reaching life end are removed from the population and a certain number of agents are randomly picked to form the next step, which mirrors the process of natural selection. For simplicity, the energy consumed by each action and the fitness used in selection process are not considered. Next, we explain each action in detail.

Mate choice A female a_i selects the best-matched male a_i as a mate from L sampled males. The sampled population consists of randomly selected L males. The female evaluates a male by calculating the hamming distance between her own preference and the traits displayed by the male. The evaluation value $P_{i,j}$ for mate choice is determined using agent a_i 's meme preference $meme_i^{pref}$ and agent a_j 's gene trait \mathcal{G}_{i}^{t} as

$$P_{i,j} = H(meme_i^{pref}, \mathcal{G}_i^t), \tag{4}$$

where H(A, B) is the hamming distance between A and B. Agent a_i prefers a_j to a_k when $P_{i,j} < P_{i,k}$. Among L males, a female a_i chooses a single mate a_j , where $P_{i,j}$ is smallest. Male choice tactics is based on a method (Kawata and Yoshimura, 2000) that is a reasonable assumption for real animals (Gibson and Langen, 1996; Widemo and Sæther, 1999). After choosing a mate, a female a_i is added to the $potential_mate_j$ list for selected male a_j . A male decides on N females randomly in the $potential_mate_j$ list after all females have selected a single mate.

A male can breed with N females per step. When the sex ratio between males and females is equal, we can explain N as the operational sex ratio (OSR) (i.e., the ratio between sexually active males and females in a population) of males. From now, we call N the OSR of the males.

Breeding Suppose that female a_i selects male a_j each other. A new agent a_l is produced as the child of a_i and a_j . This new agent a_l has the following composition.

$$a_l(sex_l, 0, (g_l^{trait}, g_l^{strategy}, meme_{DV}^{pref}),$$
 (5)

where sex_l is either male or female with an even probability; age_l is zero; genes $(g_l^{trait}, g_l^{strategy})$ are determined by genetic operations of Equation (6)(7).

$$g_l^{trait} = mutb(crb(g_i^{trait}, g_j^{trait})),$$
 (6)

$$g_l^{strategy} = mutb(crb(g_i^{strategy}, g_j^{strategy})), \qquad (7)$$

where mutb(A) is a mutate-function that reverses each bit of A with probability γ ; crb(A,B) is a cross-function that returns either A and B with an even probability. Meme is not inherited from parents. Thus, its default is $meme_{DV}^{pref}$.

A female a_i is limited to only one round of breeding for each step. On the other hand, a male a_j is limited to N round of breeding with females in his $potential_mate_j$ list for each step.

Social influence Next, the preference of a female a_i is influenced by the social environment. This means that a female a_i prefers the trait g_k^{trait} of a male a_k who is the most popular or unpopular as indicated by mate choice from other females. That is, a female a_i imitates the gene trait g_k^{trait} with her meme preference $meme_i^{pref}$. Females judge whether a male is popular or unpopular by the number of females in his $potential_mate_i$ list. The mating-strategy gene $g^{strategy}$ is encoded by a bit-string with a length of one bit. Specifically, when the mating-strategy gene $g_i^{strategy}$ of a female a_i is zero, her imitation target is the male agent whose *potential_mate* list has the least number of females in L males selected randomly from a population. Conversely, when the mating-strategy gene $g_i^{strategy}$ is one, her imitation target is the male agent whose potential_mate list has the most number of females in L sampled males.

Here, a female a_i can change her $meme_i^{pref}$ by reversing multiple bits in her bit string data to come close to the gene trait g_k^{trait} of target male a_k . The number of reversing bits is randomly spread over half the total number of all bits.

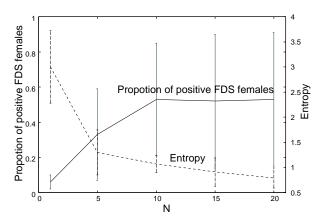


Figure 2: The average proportion of positive FDS females and the entropy of male traits after 3,000 steps at each OSR of males (N) for 40 trials.

Experiments

Next, we describe our experiments with the proposed model, where many male and female agents exist and are evolvable.

Experimental settings

All females can breed after 2[step], and select a mate described above. All parents produce two offspring for each breed. All agents are dead after 20[step] (life time). An initial population of 1,000 agents, consisting of 500 females and 500 males, is evolved from an initial state where: (1) gene g^{trait} and meme $meme^{pref}$ are encoded by bit-strings; the length of these strings is 10 bits each; (2) the initial values of genes g^{trait} and $g^{strategy}$, and meme $meme^{pref}$ are given randomly to all agents. In each step, 1,000 agents are randomly picked to form the next step. The parameterization used in these sets of simulation runs is (1) mutation rate $\gamma=0.01$; (2) simulation steps =3,000; (3) the number of sampled males L=40.

In the experiments, we focus on the transition of matingstrategy genes and the gene diversity of male traits. An alternative approach to summarizing and forecasting gene diversity can be based Shannon's entropy (Shannon, 2001). Gene diversity E is calculated as

$$E = -\sum_{i=1}^{S} p_i \log_2 p_i,$$
 (8)

where p_i is the occurrence probability of ith agents that have the same gene trait in a population and S is the number of gene traits patterns.

Results

We changed the OSR of males, N, in the range of $1 \le N \le 20$ and studied the evolution of the mating-strategy gene. Figure 2 shows the average proportion of female agents

whose mating-strategy is positive FDS and the average entropy of male traits at each N. Results showed that as N becomes larger, the proportion of positive FDS females becomes higher and comes nearer to 0.5. The entropy of male traits becomes lower as N becomes larger.

Discussion

As mentioned above, a female is added to the $potential_mate$ list of a male selected as a mate. A male can breed with N females in his $potential_mate$ list per step. When the number of females in his list exceeds N, some of the females cannot breed. The mating-strategy genes of females failing to breed then decrease at the next generation. The point here is that the mating-strategy of a female preferring a male whose $potential_mate$ list exceeds a breeding capacity (N) is disadvantageous in terms of survival.

The proportion of positive FDS females in Figure 2 becomes higher as N becomes larger. We assume this is because, as N becomes larger, 1) it is harder for the number of females in a male's $potential_mate$ list to become more than N, 2) females leave offspring more easily despite of preferring popular males, and 3) it is easy to increase the mating-strategy genes of preferring popular males; that is, the proportion of positive FDS females becomes higher. Conversely, as N becomes smaller, 1) it is easier for the number of females in male's $potential_mate$ list to become more than N, 2) females preferring popular males have a harder time leaving offspring, and 3) it is easy to decrease the mating-strategy genes of preferring popular males; that is, the proportion of positive FDS females becomes lower.

Also, as shown in Figure 2, the proportion of positive FDS females comes nearer to 0.5 as N becomes larger. This is because the mating success of females approaches 100% regardless of female mating-strategy as N becomes larger. This means that the incidence of positive or negative FDS is irrelevant.

Effect of number of males sampled for female choice

Next, we changed the number of males sampled for female choice, L, in the range of $5 \le L \le 60$, and studied the resulting evolution of the mating-strategy gene. The OSR of males (N) is fixed to one. Figure 3 shows the average proportion of female agents whose mating strategy is positive FDS and the average entropy of male traits after 3,000 steps at each L sampled males for 40 trials.

Experimental results showed that as the size of the males sampled (L) became larger, the proportion of positive FDS females became lower and the entropy of male traits became higher. This is because it was easy for the sampled males to overlap as the sample size becomes larger. Duplicate sampling caused concentration of popularity. Then, as mentioned above, it was easy for the number of females in

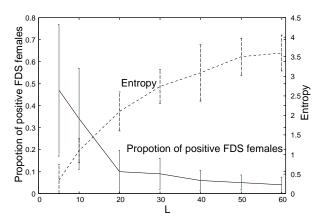


Figure 3: The average proportion of positive FDS females and the entropy of male traits after 3,000 steps at each L sampled males for 40 trials.

a male's $potential_mate$ list to exceed his capacity and the proportion of positive FDS females became smaller.

Inversely, as the size of sampled males became smaller, the proportion of positive FDS females became higher, approaching 0.5. This means that the incidence of positive or negative FDS is irrelevant, since as the size of sampled males becomes smaller, there seems to be no effect of the FDS. FDS only exhibits an effect for animals with a large enough number of sampled males.

Comparison with real animals

In this section, we compare the experimental results with real animals. According to Singh et al. (Singh and Sisodia, 2000), negative FDS has been observed in 14 kinds of animals (parasitic wasp, flour beetle, etc.) while the matechoice copying has been observed in 25 kinds of animals (sailfin molly, Japanese medaka, etc.) (Vakirtzis, 2011). Here, we treat mate-choice copying as positive FDS, since mate-choice copying has been shown to produce positive FDS (Kirkpatrick and Dugatkin, 1994).

The operational sex ratio (OSR) is an important environmental factor that influences mating behaviors (Clutton-Brock and Parker, 1992). We consider the OSR of each mating system. The OSR in monogamous species is approximately equal (Zug et al., 2001). The OSR in polyandrous species is typically strongly female biased (Schamel et al., 2004) while that in polygynous species is male biased (Jorgensen and Fath, 2008). The OSR in promiscuous species is not clear, but it might be male biased since males might have relatively higher potential reproductive rate than females. Then, we can explain the OSR of males as Polyandrous < Monogamous < Polygamous, Promiscuous.

Figure 4 shows the number of species having been confirmed to practice FDS at each mating system. White and black boxes indicate the distribution of 11 species exhibiting

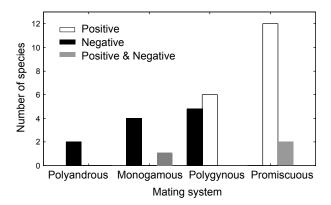


Figure 4: Number of species having been confirmed to practice FDS at each mating system. White boxes indicate positive FDS. Black boxes indicate negative FDS. Gray boxes indicate 3 species exhibiting both positive and negative FDS.

only positive FDS and 18 species¹ exhibiting only negative FDS, respectively. Gray boxes indicate the distribution of 3 species (guppy, drosophila, and humans) exhibiting both positive and negative FDS. In this figure, we can confirm that positive FDS occurs only in species whose OSR of males is large whereas negative FDS occurs in species whose OSR of males is small. This result partially matches our experimental results in Figure 2.

Several experiments (Doucet et al., 2004; Slagsvold and Viljugrein, 1999) and studies (Vakirtzis and Roberts, 2009, 2010; Dubois, 2007) have argued that mate-choice copying is highly unlikely to evolve in monogamous species due to the increasing costs of female competition. The OSR of males belonging to monogamous species is less than that of polygamous males. Our experimental results showing that as the OSR becomes smaller, the positive FDS females becomes lower, confirm these hypotheses.

On the other hand, the number of sampled males (L) of real animals is not clear. We still need to clarify whether the experimental results agree with the behavior of real animals.

Conclusion

We proposed a new model for mate choice involving genes and memes that introduces mating-strategy genes. The mating-strategy gene means that females prefer whether popular or unpopular males. We expressed female agents that have the mating-strategy gene and the meme preference influenced from other mate choice. The results of experiments using our proposed model demonstrated that as the operational sex ratio (OSR) of males becomes larger, the number of positive FDS females increases. A similar tendency was observed among real animals. We also discov-

ered that as the size of males sampled for female choice became larger, the number of positive FDS females decreased. However, we have not matched the results of this experiment against real-world animals. We need to examine the number of sampled males of actual animals and compare it with our experimental results.

We would like to point out here that despite strong evidence for the negative FDS, the evolutionary processes that account for its prevalence are not known (Hughes et al., 2013). From the results of experiments using our proposed model, we can build two hypotheses for this question. One, the OSR is not strongly male biased. Two, the sampled male size is too big.

However, mate choice in the real world is not as simple as in our model. There are a variety of factors involved in propagation, such as cost, fitness and asymmetry in the roles between males and females. We need to consider the existence of male meme traits such as decoration and mimesis. In our future work, we will improve the model based on the findings of this paper so that it better conforms to the real world.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 26730154.

References

- Clutton-Brock, T. H. and Parker, G. A. (1992). Potential reproductive rates and the operation of sexual selection. *Quarterly Review of Biology*, pages 437–456.
- Dawkins, R. (1989). The Selfish Gene. Oxford University Press.
- Doucet, S. M., Yezerinac, S. M., and Montgomerie, R. (2004). Do female zebra finches (taeniopygia guttata) copy each other's mate preferences? *Canadian Journal of Zoology*, 82(1):1–7.
- Dubois, F. (2007). Mate choice copying in monogamous species: Should females use public information to choose extrapair mates? *Animal Behaviour*, 74(6):1785–1793.
- Dugatkin, L. A. (1992). Sexual selection and imitation: females copy the mate choice of others. *American Naturalist*, pages 1384–1389.
- Gibson, R. M. and Langen, T. A. (1996). How do animals choose their mates? *Trends in Ecology & Evolution*, 11(11):468–470.
- Hughes, K. A., Houde, A. E., Price, A. C., and Rodd, F. H. (2013). Mating advantage for rare males in wild guppy populations. *Nature*, 503(7474):108–110.
- Jorgensen, S. E. and Fath, B. (2008). Encyclopedia of Ecology, Five-Volume Set, volume 1. Newnes.
- Kawata, M. and Yoshimura, J. (2000). Speciation by sexual selection in hybridizing populations without viability selection. *Evolutionary Ecology Research*, 2(7):897–909.

¹This is with the exception of four species; two species whose mating system is uncertain, one species with reversed sex roles, and one gynogenetic species.

- Kirkpatrick, M. and Dugatkin, L. (1994). Sexual selection and the evolutionary effects of copying mate choice. *Behavioral Ecology and Sociobiology*, 34(6):443–449.
- Kirkpatrick, M. and Ryan, M. J. (1991). The evolution of mating preferences and the paradox of the lek. *Nature*, 350(6313):33–38.
- Kokko, H., Jennions, M. D., and Houde, A. (2007). Evolution of frequency-dependent mate choice: keeping up with fashion trends. *Proceedings of the Royal Society B: Biological Sciences*, 274(1615):1317–1324.
- Lerena, P. (2000). Sexual preferences: Dimension and complexity. In *Proceedings of the Sixth International Conference of The Society for Adaptive Behavior*, pages 395–404.
- Mutoh, A., Kato, S., Inuzuka, N., and Itoh, H. (2010). Expression of fashion in female preferences for a mate by conformity and differentiation genes. In *Artificial Life XII*, *Proceedings of the Twelfth International Conference on the Synthesis and Simulation of Living Systems*, pages 846–851.
- Schamel, D., Tracy, D. M., and Lank, D. B. (2004). Male mate choice, male availability and egg production as limitations on polyandry in the red-necked phalarope. *Animal behaviour*, 67(5):847–853.
- Servedio, M. R. and Kirkpatrick, M. (1996). The evolution of mate choice copying by indirect selection. *American Naturalist*, pages 848–867.
- Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55.
- Singh, B. N. and Sisodia, S. (2000). Frequency-dependent selection: Minority male mating advantage in drosophila. *Current Science (Bangalore)*, 78(2):141–150.
- Sirot, E. (2001). Mate-choice copying by females: the advantages of a prudent strategy. *Journal of Evolutionary Biology*, 14(3):418–423.
- Slagsvold, T. and Viljugrein, H. (1999). Mate choice copying versus preference for actively displaying males by female pied flycatchers. *Animal Behaviour*, 57(3):679 686.
- Tokuhara, S., Mutoh, A., Kato, S., and Itoh, H. (2005). A sexual selection model with genes and memes. *Proc. of the 7th International Conference on Artificial Evolution(EA 2005), on CD-ROM.*
- Vakirtzis, A. (2011). Mate choice copying and nonindependent mate choice: a critical review. In *Annales Zoologici Fennici*, volume 48, pages 91–107. BioOne.
- Vakirtzis, A. and Roberts, S. C. (2009). Mate choice copying and mate quality bias: different processes, different species. *Behavioral Ecology*, 20(4):908–911.
- Vakirtzis, A. and Roberts, S. C. (2010). Nonindependent mate choice in monogamy. *Behavioral Ecology*, 21(5):898–901.
- Westneat, D. F., Walters, A., McCarthy, T. M., Hatch, M. I., and Hein, W. K. (2000). Alternative mechanisms of nonindependent mate choice. *Animal Behaviour*, 59(3):467–476.

- Widemo, F. and Sæther, S. A. (1999). Beauty is in the eye of the beholder: causes and consequences of variation in mating preferences. *Trends in Ecology & Evolution*, 14(1):26–31.
- Witte, K. and Noltemeier, B. (2002). The role of information in mate-choice copying in female sailfin mollies (poecilia latipinna). *Behavioral Ecology and Sociobiology*, 52(3):194–202.
- Zug, G. R., Vitt, L. J., and Caldwell, J. P. (2001). Herpetology: an introductory biology of amphibians and reptiles. Academic Press.

Evolution of Communication in Mate Selection

Aditya Rawal¹, Janette Boughman² and Risto Miikkulainen³

^{1,3}University of Texas, Austin, TX ²Michigan State University, East Lansing, MI aditya@cs.utexas.edu

Abstract

A computational study is conducted to evaluate the hypothesis that mate selection is the evolutionary origin of communication. A population of neural networks is evolved for two cooperative tasks - mate selection and prey capture. Simple codes developed for mate selection serve as an effective stepping stone for prey capture but not vice versa. Mate selection followed by an additional prey capture task is also easier to evolve than both together from the beginning. This result suggests that mate selection may be a first step in the evolution of general communication systems in nature.

Introduction

An important scientific question is to understand the evolutionary origin of language. Communication in nature is believed to have emerged through several stages (Corballis, 2011; Searcy and Nowicki, 2010; Bradbury and Vehrencamp, 2011). Initially, non-signals (neutral traits/natural cues) like breathing patterns and urination were ritualized as courtship signals and territorial markings, respectively (Smith, 1997; Hinde, 1997). Next, complex social and environmental pressures required individuals within a group to cooperate and compete in diverse environments (Nowak and Krakauer, 1999; Corballis, 2011). Such interactions gradually lead to more complex signals, and eventually to a protolanguage (Bickerton, 1990). Subsequently, signals developed for a specific evolutionary purpose were exapted and used for other functions (Logan, 2008; Gould, 2002). These steps are believed to have been crucial in the evolution of human language.

Without direct evidence, it is difficult to verify that these steps indeed took place, and to identify the actual behaviors involved. However, simulating language evolution using modern computer science tools can be insightful. Through simulated experiments, this paper evaluates the hypothesis that mate selection is the evolutionary origin of communication. Because mate selection is essential in sexual organisms, is closely related to fitness, and is a social interaction, it is a compelling candidate. During mating, communicative signals emerge naturally to help identify suitable mates. As

soon as such signaling is possible, it may be exapted to serve other social behaviors as well.

In this paper, a series of computational evolution experiments are performed to simulate the origins of mating signals and their exaptation for other social tasks. Communication is evolved in a population of artificial agents in two cooperative tasks: mate selection task and prey capture. In nature, communication during the mating process is used to discern suitable partners and successfully realize reproduction to produce fit offspring (Hauser, 1997). Analogously, during simulated mating, a pair of randomly selected agents from the population exchange signals and determine their compatibility. Initially, this process is based on direct display of traits. In the second phase, simple codes emerge that encode these traits. These codes can be seen as vocal/visual gestures used by individuals in nature.

In the third phase, the successful population from the mate selection experiment is evaluated in an additional task - prey capture. This task requires a more sophisticated communication mechanism on the part of agents in order to time their attack to capture the prey. They can leverage the communication abilities developed for mate selection and quickly achieve the prey capture goal. This adaptation turns out to be computationally faster than evolving communication for prey capture first, or evolving for both mate selection and prey capture at the same time. The experiments thus demonstrate that complex signaling in tasks like prey capture could have been exapted from a simpler task such as mate selection. The computational results thus support the theory that mate selection is the origin of communication.

Prior Work

Although several prior computational studies demonstrated the evolution of communication in intelligent agents, most did not focus on possible biological origins of this process. The environments and tasks were carefully crafted so that communication was necessary for the agents to be successful, and the focus was simply on developing a common communication code (Nolfi and Mirolli, 2010; Steels, 2003, 2005; Tuci, 2009; Werner and Dyer, 1990; Rawal et al.,

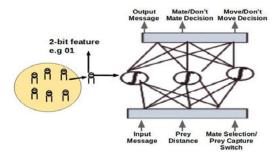


Figure 1: **Population of Agents** - Each agent in the population consists of a 2-bit feature and a feedforward neural network. The feature is required during mate selection to determine agent compatibility, and the network implements the behavior of the agent.

2012).

(Mirolli and Parisi, 2010) studied the role of kin-selection in the emergence of the first signalers and receivers. (Quinn, 2001) demonstrated how simple behavioral cues can be selected by evolution and subsequently used as signals. (Greeff and Nolfi, 2010) evolved implicit and explicit signaling between a team of foraging robots, demonstrating that . Importantly, the individuals in these experiments are homogenous i.e a single genotype is cloned to generate a team. This design makes the task of evolving a common code easier. In contrast in nature, although communicating individuals do share a common code (i.e they have a common agreement on the meaning of the signals) their individual characteristics like size, strength and speed vary. Therefore, a heterogenous population of neural networks is evolved in this paper. An important result is that even with such heterogeneity, a common encoding scheme emerges.

(Mitri et al., 2010) highlighted the importance of group selection in the evolution of honest signaling. Similarly, the experiments designed in this paper are cooperative i.e the team fitness reward is equally distributed between the participants.

The paper is organized as follows. The experimental setup is described first, including the network architecture and the two tasks, mate selection and prey capture. This section also explains the two cooperative tasks - mate selection and prey capture. The results of the five experiments - explicit and implicit communication in mate selection and the three ways of evolving communication for the two tasks - are then presented. The computational and biological insights and future extensions to competitive mating are discussed in the end.

Experimental Setup

A common network architecture is evolved to perform both the tasks of mate selection and prey capture, as described below. Each individual agent consists of a neural network and a 2-bit binary feature (See Figure 1). The 2-bit feature is

Features	0 0	01	10	11
0 0	1	0	0	1
0 1	0	1	1	0
10	0	1	1	0
11	1	0	0	1

Figure 2: **Feature-Compatibility Matrix** - Compatibility of any two agents is determined by the compatibility of their 2-bit features. An entry of one in this matrix indicates that the two features are compatible and a zero indicates that they are not.

used during the mate selection task to determine whether the agents are compatible or not. The neural network controls the behavior of the agent during simulation.

Neural Network Architecture

An agent's neural network consists of a single hidden layer with 10 hidden neurons and four output neurons. The first two output neurons are used as the output message of the agent. The third output is used during mate selection task to indicate mating decision i.e mate/dont-mate. The fourth output is used during prey capture task to indicate agent movement decision i.e move/don't-move. All nodes have a sigmoidal non-linearity and therefore the output value of each node is a real number ranging between 0 and 1. The output node values are rounded off to 0/1 before being used.

In each experiment, there are at least four input neurons - two messaging inputs for receiving messages from other agents, one prey-distance input for sensing prey (integer values between [0-2] and set to -1 during mate selection) and one task-switch input to indicate the type of current task (1 during mate selection and 0 otherwise). In Experiment 1, there are two additional network inputs that provide the agent its own feature value. The network architecture is fixed over the course of evolution. At the start of simulation, a population of 64 agents is generated with random neural network weights and a random 2-bit feature. During each evolutionary generation, 3000 trials of a given experiment are performed on a pair of randomly selected networks. Binary tournament selection is used to select parent population. To generate the offspring population, the neural network weights of the parent population are mutated with a probability of 0.4 by adding a Cauchy-distributed random value to it. Note, that the 2-bit feature value associated with each agent is not mutated and is directly transferred from parent to offspring.

Two cooperative tasks are used to evaluate the population - mate selection and prey capture as described below.

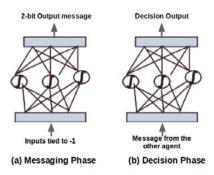


Figure 3: **Agent Behavior** - The agents implement their mate-selection and prey-capture behavior through two network activations: (a) Messaging Phase - The network input nodes dedicated to receive messages are tied to a constant value of -1. In mate selection, they output a code for their 2-bit feature. In prey capture, they output a 2-bit code for distance to the prey. The 2-bit message is then given to the other network as input. (b) Decision Phase - The networks are activated and their output decision node value is rounded off to give a 0/1 decision (mate/don't mate or move/don't move. In this manner, the networks communicate their states first and then decide upon a proper action to take.

Mate Selection Task

During the mate selection process in nature, males often try to communicate their characteristics through simple cues, gestures and more sophisticated signaling (Stoffer and Walker, 2012; Johansson and Jones, 2007). These signals convey the fitness of the individual (and therefore fitness of a future offspring) and thus influence the prospects of mating with the female.

The mate selection task emulates this process. A 2-bit feature associated with an agent represents its individual characteristic in this artificial setting. For simplicity, the population is not divided into males and females. Instead, two agent networks are randomly selected from the population and they exchange messages to make a mate/dont-mate decision. The ground truth for this mating decision is predefined in a randomly generated feature-compatibility matrix. One example of such a matrix is shown in Figure 2. A 1 in the matrix indicates that the two agents are compatible while a 0 indicates that they are not. From a biological perspective, compatibility between two agents can be viewed as the fitness of their offspring i.e if two highly compatible agents decide to mate, their offspring will have a higher fitness than the offspring of two incompatible agents.

Such a mating task is cooperative in that none of the agents receive fitness increment unless both of them make the correct mating decision. Both agents receive a fitness increment of one for each correct mating decision. For example, feature values 01 and 10 in Figure 2 are compatible and therefore two agents with these features receive a fitness

increment of one if both decide to mate and zero if otherwise.

During this task, the task-switch and prey-distance inputs are set to constant values of 1 and -1, respectively. Each trial consists of two phases (Figure 3). In the first phase (defined as the messaging phase), the two neural networks are activated (with messaging input tied to constant value of -1) and the first two output nodes are sampled for their values. The two output real values are then rounded off to obtain a 2-bit output message. In the second phase (the decision phase), this 2-bit output message from one network is given as messaging input to the other network. After a second network activation, the value at the output mating decision node is rounded off to generate a mate/dont-mate decision.

Note that in the current simulations the two paired networks do not undergo actual mating (i.e crossover) based on their mating decision. Instead, offspring is generated through mutations of most fit parents. This simplification helps focus the study on the question of how mate selection may act as an origin for communication. However, to this end it is essential that the agents are not artificially constrained to generate any specific message type during the messaging phase. The only feedback the agents receive (indirectly, through fitness) is based on the correctness of their mating decision.

An additional constraint is added to ensure that each feature is equally represented in the population over the course of evolution. The initial population of size 64 is divided into four sub-groups of size sixteen each - one group for one feature type. After all networks have been evaluated in each generation, a binary tournament is performed on each feature group separately to obtain sixteen offspring. Since the parent feature is directly transferred to offspring and is unaffected by mutation, the feature representation in population remains fixed at sixteen. This restriction prevents any feature in the population from becoming dominant (which would remove the need to communicate).

Prev Capture Task

Prey capture in nature often involves cooperation and communication on the part of agents. This task simulates such a scenario with a fixed prey in a discrete, deterministic world. At the start of each trial, two agents (randomly selected from the population) are positioned at random distances apart from the prey (Figure 4). The agent agents can sense the distance to the prey but cannot sense the distance to the other agent. They have two possible actions - either move towards the prey or stay in their current position. With each agent move towards the prey, the agent-prey distance decreases by one. The prey is considered captured when the two agents pounce on the prey in the same time step. In order to be successful, the agents need to communicate when they are ready for the capture.

The task-switch input neuron is set to a constant value

of 0 to indicate prey capture task, and the prey distance inputs to the proper value to indicate distance. Similarly to the mate selection task, each time step in a trial is divided into two phases - messaging phase and decision phase. The output movement decision node of the network determines the agent action: move/dont-move. agent agents receive a fitness increment of one on capturing the prey and zero otherwise. Again, agents are not constrained to generate a specific message type but must evolve one that works.

Measuring Success

In each experiment, data is collected over 150 runs and averaged. Since fitness distribution is cooperative for each task (i.e in a given trial, a pair of individuals receive fitness of one only if both make correct predictions), it is very difficult to achieve 100% population success. Therefore, a population is deemed successful if at least 25% of its individuals have an average fitness of greater than 75% of the maximum fitness. This success criteria ensures that a significant fraction of population develops common communication code to solve the task. The total number of generations required to evolve a successful population is used as a quantitative measure to compare different experiments.

The next section describes the experiments in detail.

Experiments

The first two experiments demonstrate that a common code does evolve in mate selection. Experiments 3-5 then evaluate the hypothesis that mate selection is an effective first step in evolving a more general communication system.

Experiment 1 - Mate Selection with Feature Inputs

In the first experiment, the population is evaluated in the mate selection task. Each agent (in the randomly selected pair) is explicitly provided its own 2-bit feature as input during the messaging phase. The 2-bit message generated at its network output is then given as input to the other agent in the decision phase. In this setting, the neural network thus has four active input nodes - 2-bits for agent feature and 2-bits for sensing the output message of the other agent.

On average after 11 generations, the best agent pair evolve a simple signaling system where they send their exact input feature as a message during the messaging phase. These messages are similar to the explicitly visible characteristic of an individual in nature - for example its size and strength. The messages thus do not represent a communication code; they are simply a direct expression of the underlying characteristic. They serve as a baseline for evolving a communication code.

Experiment 2 - Mate Selection with Hidden Features

In the second experiment the setup is modified to allow an actual communication code to emerge. During the messag-

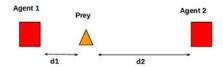


Figure 4: **Prey-capture task** - There is one static prey and two agents positioned random distances ($0 \le d1 \le 2$, $0 \le d2 \le 2$)apart from the prey. At each time step, the agents can either stay still or move one distance unit towards the prey. To be successful, they are required to capture the prey by stepping on its location at the same time step. The agents can sense the prey distance but not the distance to the other agent. Thus, they need to develop a communication system to coordinate their movements.

ing phase, the 2-bit input feature is hidden from the agent i.e not given to the neural network as input. Evolution has to figure out this feature and develop a code for communicating it. Thus, only messaging inputs of the network are active during the decision phase.

On average, after 31 generations, the agent pairs develop to send an encoding of their input features during messaging phase. These encoded messages provide the necessary information to distinguish compatible agents from the not-compatible ones. In certain cases, the encodings are shared between compatible individuals. The encoding evolved for the feature-compatibility matrix shown in Figure 2 is one such case. Since feature values '00' and '11' are compatible with both themselves and each other, evolution discovers a shared message (e.g '00') to recognize these two features (exact code values vary across different runs). In this manner, a true communication system evolved to express the features hidden from the agents. This system can be exapted to other tasks, as shown below.

Experiment 3 - Mate Selection followed by Mate selection and Prey Capture

This experiment simulates how simple communication that was initially evolved for mate selection can be exapted for the more complex prey capture. The population is first evolved for the mate selection task without feature inputs (Experiment 2). The successful population is then evolved in two tasks - both mate-selection and prey-capture together.

During the mate selection task, prey-distance input is tied to a constant of -1 and task-switch input is set to 1. The feature-compatibility matrix shown in Figure 2 is used. Once a quarter of population achieves an average fitness of more than 75% of the maximum fitness in the mate-selection (criteria of success as described earlier), an additional prey capture task is introduced. In each generation, the population is separately evaluated in the two tasks. Fitness of an individual in the population is its average fitness in the two tasks. The input/output communication channels developed

during the mate selection task are reused during prey capture task. The experiment terminates once the population achieves the average fitness goal for both the tasks (i.e quarter of population with an average fitness greater than 75%). During the course of evolution, feature representation in the population is kept constant (as described in the mate selection task).

During the prey capture task, the prey distance input to the network is set according to distance and task-switch input is set to 0. Agents are positioned at random distances from the prey. The maximum initial agent-prey distance is two steps and the minimum distance is one step. Agents can choose to move/dont-move towards the fixed prey by activating their movement output node.

The evolved communication codes become quite sophisticated. When the first agent is at a distance of more than one from prey, it sends a code (e.g '00') signaling that it is moving and has yet to reach next to the prey. At a distance of exactly one step from the prey, it changes its output (e.g '11') indicating that it is now ready to capture the prey. The second agent ignores these signals from the first agent while it is not ready for prey capture (i.e it is at a distance of more than one step from prey), and instead it moves towards the prey. Once the second agent reaches a distance of exactly one step from prey, it starts paying attention to the signal from the first agent. If the first agent is ready for prey capture (i.e it signals '11'), both agents move to the prey location to capture it. Otherwise, the second agent selects the dont-move option and waits until the first agent is ready (outputs '11').

Solution to the first task (mate selection) evolves on average in 31 generations (same as in experiment 2). It takes on average 55 additional generations to evolve a population that solves both tasks. Thus on average a total of 86 generations are required to solve the two tasks together completely. The entire profile of this process is shown in Figure 5

Interestingly, the codes used during mate selection task are reused during the prey capture task. For example, one pair of agents used the code '11' to communicate their own feature in the mate selection task. After evolving in the prey capture task, their descendants used this same code to indicate readiness for prey capture. In addition, they evolved a second signal ('01') to indicate that they were moving towards the prey. In this manner, the mate selection gave them a starting point that was modified for the new task.

Experiment 4 - Mate Selection and Prey Capture Coevolution

In this experiment, communication is evolved for mate selection and prey capture simultaneously. Starting from the first generation, each agent is evaluated in both tasks, and their fitness is averaged across tasks. Thus, a population is deemed successful if it achieves the average fitness goal for both tasks. Feature representation in the population is kept

constant by performing binary tournament selection on each feature sub-group separately, as before.

A successful population evolves on average in 123 generations. As shown in Figure 5, coevolution of both tasks tends to take longer than when mate selection is evolved first. A successful population is evolved on average in 123 generations (vs. 86 generation in Experiment 3; p < 0.066). Thus mate selection serves as a useful stepping stone for other tasks.

Experiment 5 - Prey Capture followed by Mate Selection and Prey Capture

This experiment is similar to Experiment 3, except the roles of mate selection and prey capture are reversed. The population is first evolved to solve the prey capture task and then to solve both mate selection and prey capture. The input/output communication channels developed during the prey capture task are exapted to the mate selection task.

Initially, during the prey capture task, the agent features are irrelevant (since agent compatibility information is not used). Therefore, binary tournament is performed on the whole population irrespective of the feature type. Once a successful population evolves for the prey capture task (again, a quarter of the population performing at 75%), mate selection is introduced as a second task for evaluation. Since mate selection requires information about agent compatibility, 2-bit features are assigned randomly to individuals in the population at that point. Subsequently, binary tournament is performed on each feature group separately as usual in mate selection.

A successful population evolves on average in 183 generations. As shown in Figure 5, exaptation from prey capture to mate selection takes longer than both exaptation from mate selection to prey capture (Experiment 3) and their coevolution (Experiment 4); these differences are statistically significant p < 0.0001). This result suggests that it is not simply the incremental nature of exaptation that matters but the actual steps; mate selection is a better stepping stone for building a general communication system than prey capture is.

Discussion

As shown in Figure 5, it is easiest to evolve a population for the two ecological tasks of mate selection and prey capture, if it is first evolved to solve the mate selection task. A prerequisite for communication includes the ability to output meaningful signals and interpret the received input signals (Mirolli and Parisi, 2010). The mate selection task builds an initial structure to support this mechanism. The signals that evolve for mate selection are simple and fundamental: they map a single state to a single symbol. Once created, this communication mechanism can be be co-opted for more difficult tasks like prey capture, where multiple states must be mapped to the same symbol, or multiple symbols have the

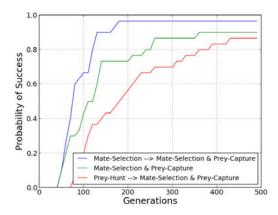


Figure 5: **Probability of success throughout evolution** This graph shows how likely evolution is to discover a successful population (where a quarter of the population is successful 75% of the time in the two tasks of mate selection and prey capture) by a given generation. The most effective method is to evolve mate selection first, followed by additional prey capture. Evolving both at once from the beginning is weaker, but still stronger than evolving for prey hunting first followed by mate selection. Thus, communication evolved for mate selection makes it easier to evolve communication for another task. This result suggests that mate selection is a likely first step in the evolution of communication.

same meaning. Indeed, networks evolved initially for mate selection end up using significantly fewer symbols than coevolved or initial prey-capture networks (with p < 0.001; Figure 6). From a computational perspective, the mate selection task thus provides a better stepping stone to construct general communication than prey capture.

Future Work

The mate selection process in these experiments was cooperative in that both partners aimed at determining compatibility, and there was no conflict of interest. While such mate selection is seen in nature, competitive mate selection is more common (Andersson, 1994). That is, one of the genders, usually female, chooses the mate, and the other gender, usually male, tries to convince that he is a good choice. Competitive mate selection likely originates from the asymmetry in gender roles: Females try to maximize quality and males quantity of their mates. Competitive mating thus leads to behaviors where males may try to communicate dishonestly, exaggerating their quality, and females try to estimate what the males' actual quality is. On the other hand, most other communication between animals is by necessity cooperative. For instance in hunting, the individuals need to communicate honestly in order to be effective. An interesting

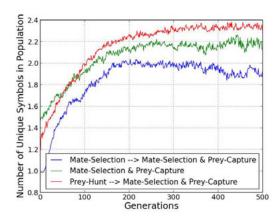


Figure 6: The Number of Communication Symbols Evolved - With prey capture, there are many states when the agent is not ready, and therefore some redundant symbols are likely to evolve to signal these states (e.g 2.4 symbols on average). In contrast, mate selection often results in a one-to-one mapping from states to symbols. Therefore, the final communication system built on mate selection is likely to be simpler (about 2.0 symbols on average) and therefore easier to discover than with the alternatives. The population success criterion was increased from 75% to 90% of maximum fitness in this comparison to reduce variance.

scientic question therefore is: How can an eventually mostly cooperative communication system be built on a competitive origin? This question will be studied in future extensions of the simulations described in this paper.

Conclusion

This paper evaluated computationally the hypothesis that mate selection is the origin of communication in nature. That is, communicative signals emerge naturally to help identify suitable mates; as soon as such signaling is possible, it is exapted to serve other social behaviors, such as prey capture. Computational simulations verified this hypothesis in a simplified setting where direct displays of traits evolved to coded communication of traits, and were then exapted to use in coordinating a prey capture. This sequence turned out to be faster than evolving prey capture first or both together from the beginning, suggesting that mate selection may indeed be an effective stepping stone to construct general communication systems.

Acknowledgements

This research was supported in part by NSF grants DBI-0939454 and IIS-0915038, and in part by NIH grant R01-GM105042.

References

- Andersson, M. (1994). Sexual Selection. Princeton University Press.
- Bickerton, D. (1990). Species and Language. Univ. of Chicago Press, Chicago.
- Bradbury, J. W. and Vehrencamp, S. L. (2011). *Principles of Animal Communication*. Sinauer.
- Corballis, M. C. (2011). *The recursive mind: the origins of human language, thought, and civilization.* Princeton: Princeton University Press.
- Gould, S. (2002). *The Structure of Evolutionary Theory*. Harvard University Press.
- Greeff, J. and Nolfi, S. (2010). Evolution of implicit and explicit communication in mobile robots. Berlin: Springer.
- Hauser, M. D. (1997). The Evolution of Communication.
- Hinde, R. (1997). Animal Behavior.
- Johansson, B. and Jones, T. (2007). The role of chemical communication in mate choice. *Biol Rev Camb Philos Soc.*, 82:265–89
- Logan, R. (2008). The Extended Mind: The Emergence of Language, the Human Mind, and Culture. University of Toronto Press
- Mirolli, M. and Parisi, D. (2010). Producer Biases and Kin Selection in the Evolution of Communication: How the Phylogenetic and the Adaptive Problems of Communication Can Be Solved. Springer Verlag.
- Mitri, S., Floreano, D., and Keller, L. (2010). *Evolutionary Conditions for the Emergence of Communication*. Springer Verlag.
- Nolfi, S. and Mirolli, M. (2010). *Evolution of Communication and Language in Embodied Agents*. Berlin: Springer.
- Nowak, M. and Krakauer, D. (1999). The evolution of language. *Proceedings of the National Academy of Science*, 6.
- Quinn, M. (2001). Evolving communication without dedicated communication channels. In *Advances in Articial Life: 6th Eu-ropean Conference*, page 357366.
- Rawal, A., Rajagopalan, P., Miikkulainen, R., and Holekamp, K. (2012). Evolution of a communication code in cooperative tasks. In *Artificial Life*. MIT Press, Cambridge, MA.
- Searcy, W. A. and Nowicki, S. (2010). The Evolution of Animal Communication: Reliability and Deception in Signaling Systems. Princeton University Press.
- Smith, M. J. (1997). The Theory of Evolution.
- Steels, L. (2003). Evolving grounded communication for robots. *Trends in Cognitive Sciences*, 7:308312.
- Steels, L. (2005). The emergence and evolution of linguistic structure: From lexical to grammatical communication systems. *Connection Science*, 17:213230.
- Stoffer, B. and Walker, S. (2012). The use of multimodal communication in mate choice decisions by female house crickets, acheta domesticus. *Animal Behavior*, 83:11311138.

- Tuci, E. (2009). An investigation of the evolutionary origin of reciprocal communication using simulated autonomous agents. *Biological Cybernetics*, 101:183199.
- Werner, G. M. and Dyer, M. G. (1990). Evolution of communication in articial organisms. In *Proceedings of the Workshop on Articial Life*, page 659687. Reading, MA: Addison-Wesley.

Constrained genetic architecture promotes cooperation

Antoine Frénoy¹, François Taddei¹, and Dusan Misevic^{1*}

¹Center for Research and Interdisciplinarity, INSERM U1001, Université Paris Descartes, Sorbonne Paris Cité, France *corresponding author: dule@alife.org

Abstract

Cooperation in nature often has direct costs but only indirect benefits. Kin and group selection theories comprehensively address its evolutionary origins, but our knowledge of the precise genetic mechanisms that prevent cheater invasion and maintain cooperation is incomplete. Here we review our published work on cooperation in Aevol, an agent-based, in silico genomic platform used to evolve and study populations of digital organisms that compete, reproduce, and cooperate by secreting a public good. Motivated by the observation of phenotypically identical individuals who had radically different evolutionary fates, we recorded and compared gene locations, effectively performing bio-inspired genomics analyses of our digital organisms. We found that the association between metabolic and secretion genes (promoter sharing, overlap via frame shift or sense-antisense encoding) was characteristic for populations with robust, stable cooperation. Such architecture arose de novo during the evolution of cooperation, but only when producing the public good was costly. Effectively, cooperation evolved to be protected and maintained through constrained, entangled genetic architecture. Beyond confirming the importance of second-order selection, we uncover a novel genetic mechanism for the evolution and maintenance of cooperation. Our results suggest a method to limit the evolutionary potential of synthetically engineered organisms, in order to reduce the change or loss of synthetic gene circuits, a major issue in synthetic biology.

Background and Introduction

The evolution of cooperation in microbial populations is a fascinating, rich and controversial evolutionary problem (West et al. 2006). Evolutionary explanations of cooperation are constantly being improved and refined by a host of mathematical tools such as game theory and meta-population models (Lehmann and Keller 2006). However, these methods typically do not distinguish between genotypes and phenotypes and are unable to investigate the structure of genomes that encode the cooperative traits. Microbial studies have already hinted at potential mechanisms by which genetic architecture can affect cooperation, including gene coregulation and pleiotropy (Foster et al. 2004; Dandekar et al. 2012). Here we review our published work (Frénoy et al. 2013) that examines two specific types of genomic architecture: (1) operon sharing, when secretion and metabolism genes are on the same operon and share a promoter and a terminator, and (2) overlap, the base-pair sharing between metabolic and secretion genes possible when genes are in different reading frames or on different DNA

strands. Well described in viruses, overlaps in bacteria are existent but rare. Similar to operons, they are thought to be caused by strong maximum genome size constraints or selection for co-regulation (Normark et al. 1983). However, overlaps have not themselves been studied as an evolutionary constraint, which we do here, in the context of cooperation.

Methods

In our work we used the in silico experimental evolution system Aevol (Batut et al. 2013), heavily inspired by microbial genetics. The implementation of an Aevol digital organism allows for continuous metabolic and cooperating phenotypes. For example, instead of the classical binary cheat/cooperate behavior, there is an infinite number of possible cooperation phenotypes. Individuals live on a grid and when they evolve cooperation, it is via a secreted public good molecule that diffuses and degrades (Misevic et al. 2012). The public good is costly to secrete but may benefit any neighboring organisms. Rules for transcription, translation and protein synthesis govern the complex mapping of a double-stranded binary string (genotype) to phenotype to fitness. Phenotypically similar or even identical individuals can have different genotypes, thus also having different evolvability, robustness, and evolutionary fate (Frénoy et al. 2012). All these properties of Aevol enable the two sets of evolutionary experiments we performed, in which genetic architecture constraints relating to cooperation were both observed and quantified. We first analyzed the dynamics of cooperation loss due to increased cost of secretion (cheater invasion experiments), and then evolved cooperation from non-secreting ancestors, under different costs (de novo evolution of genetic links experiments). To quantify genetic architecture, we identified instances of all operon sharing and overlap between genes. For each of the evolved cooperators we analyzed the architecture of all its secretion genes and classified them in different categories based on presence or absence of gene overlap and operon sharing. Additionally, we constructed millions of random mutants and measured the effect of mutations on both fitness and secretion to confirm the phenotypic effect of the constrained genetic architecture.

Results and Discussion

All experimental results supported the hypothesis of cooperation maintenance via constrained genetic architecture.

(1) Cheater invasion experiments. Using a bank of 50 evolved cooperators, we seeded new populations that continued evolving under a high secretion cost. Although the starting individuals had comparable level of secretion, under the new cost regime, the populations greatly varied in the resistance to cheater invasion. In replicate experiments, some populations consistently preserved low levels of secretion, while others always lost all secretion genes. The abundance of promoter sharing or overlapping between metabolic and secretion genes was a good predictor of cooperation maintenance. Mutational analysis confirmed that the constrained genetic architecture resulted in a greater proportion of cooperation-destroying mutations also having a direct negative fitness effect, than non-constrained architecture. Secretion-decreasing mutations evolutionary dead-ends, resulting in evolvability suppression. possible spontaneously-evolving mechanism maintenance of far-sighted traits (Altenberg 2005). Interestingly, the mutation analysis was a good predictor of short-term evolution, while genetic architecture was better at explaining long-term evolutionary fate of cooperation.

(2) De novo evolution of genetic links experiments. In the second set of experiments, populations starting from non-cooperators evolved under different secretion cost regimes. Cooperators evolved to employ the constrained, protective encoding, and more so when the cooperation cost was high (Figure 1). This confirms that the entangled genetic architecture is subject to selection because of its protective effect on cooperation.

In the work reviewed here, we provide a new set of potential explanations for the evolution of operons and overlaps, both important building blocks of life. Operons and overlaps may help protect genes that are at risk of removal because of a short-term cost and in spite of the long-term benefit they may provide. In the case of cooperation, genetic architecture constraints may be highly relevant to a better understanding of the much-studied siderophore-mediated cooperation in *P. aeruginosa*, where cooperative as well as metabolic traits are under the control of the same quorum sensing mechanism (Dandekar et al. 2012). However, beyond

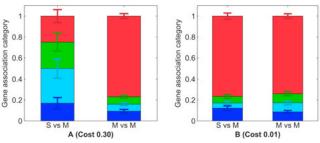


Figure 1. Genetic associations after *de novo* cooperation evolution at different secretion cost (from (Frénoy et al. 2013)). We quantified operon sharing and overlaps between secretion genes and metabolic genes (S vs M) and between metabolic genes and other metabolic genes (M vs M). Four classes of gene pairs were identified: sharing an operon without overlapping (dark blue), overlapping and sharing an operon (light blue), overlapping without sharing an operon (green), not sharing an operon nor overlapping (red). Error bars represent standard error of the mean.

explaining evolutionary outcomes in the field of cooperation, the mechanism we described here could be actively used to prevent mutations from removing the genes introduced into bio-engineering organisms, a major problem in the field of synthetic biology (Sleight et al. 2010; Yang et al. 2013; Renda et al. 2014). We have already designed tools that would allow manipulation of digital genomes in a way that would create or eliminate constrained architecture without changing the phenotype. In ongoing work, we are experimentally testing the effect of overlap on the maintenance of costly genes in synthetic *E. coli* strains, effectively connecting *in silico* and *in vitro* research to practical applications.

References

- Altenberg L. 2005. Evolvability suppression to stabilize far-sighted adaptations. Artificial Life 11:427-443.
- Batut B, Parsons DP, Fischer S, Beslon G, and Knibbe C. 2013. In silico experimental evolution: a tool to test evolutionary scenarios. BMC Bioinformatics 14.
- Dandekar AA, Chugani S, and Greenberg EP. 2012. Bacterial quorum sensing and metabolic incentives to cooperate. Science 338:264-266
- Foster KR, Shaulsky G, Strassmann JE, Queller DC, and Thompson CRL. 2004. Pleiotropy as a mechanism to stabilize cooperation. Nature 431:693-696.
- Frénoy A, Taddei F, and Misevic D. 2012. Robustness and evolvability of cooperation. Pp. 53-58 *in* C Adami, DM Bryson, C Ofria, and RT Pennock, eds. Proceedings of Artificial Life 13. MIT Press.
- Frénoy A, Taddei F, and Misevic D. 2013. Genetic architecture promotes the evolution and maintenance of cooperation. PLoS Computational Biology 9:e1003339.
- Lehmann L and Keller L. 2006. The evolution of cooperation and altruism a general framework and a classification of models.

 Journal of Evolutionary Biology 19:1365-1376.
- Misevic D, Frénoy A, Parsons DP, and Taddei F. 2012. Effects of public good properties on the evolution of cooperation. Pp. 218-225 *in* C Adami, DM Bryson, C Ofria, and RT Pennock, eds. Proceedings of Artificial Life 13. MIT Press.
- Normark S, Bergstrom S, Edlund T, Grundstrom T, Jaurin B, Lindberg FP, and Olsson O. 1983. Overlapping genes. Annual Review of Genetics 17:499-525.
- Renda BA, Hammerling MJ, and Barrick JE. 2014. Engineering reduced evolutionary potential for synthetic biology. Molecular BioSystems.
- Sleight SC, Bartley BA, Lieviant JA, and Sauro HM. 2010. Designing and engineering evolutionary robust genetic circuits. Journal of Biological Engineering 4:12-12.
- West SA, Griffin AS, Gardner A, and Diggle SP. 2006. Social evolutionary theory for microorganism. Nature Reviews Microbiology 4:597-607.
- Yang S, Sleight SC, and Sauro HM. 2013. Rationally designed bidirectional promoter improves the evolutionary stability of synthetic genetic circuits. Nucleic Acids Research 41.

Coevolution in Hide and Seek: Camouflage and Vision

Kyle I. Harrington¹, Jesse Freeman¹, and Jordan Pollack¹
¹DEMO Lab, Brandeis University, Waltham, MA 02453
kyleh@cs.brandeis.edu

Abstract

Predator-prey interactions are one of the most common coevolutionary dynamics in Nature. We consider a model of the coevolution of prey appearance and predator vision, where a successful result is visually apparent. While using a neurophysiologically-based model of vision and a rich developmental process for prey patterning, we show that predatorprey coevolution can maintain engagement. Backgrounds with large regional differences generally lead to prey that appear as mixtures of the regions. Finally, we find that engagement between predators and prey is supported by greater background complexity.

One of the most visually-striking phenomena in predatorprey coevolution is prey crypsis, the ability of prey to avoid detection by predators. Chameleons and cuttlefish take this behavior to the extreme and physically alter their pigmentation to match their environment, which can even be realized synthetically in robots (Morin et al., 2012). However, prey crypsis is often manifested as static pigmentations, such as stationary Turing patterns (Turing, 1952), that are selected for being advantageous in particular environments. Inspired by Bond and Kamil (2002), where blue jays are used to interactively evolve moth phenotypes, we study the effect of background complexity on the coevolution of prey appearance and predator vision.

There is an intimate coevolutionary relationship between predator vision and prey appearance. Visual systems are generally adapted for stimuli that exert selective pressure on the organism, such as food, prey, predators, and mates. Even when visual systems have adapted to attenuate visual signals from their relevant stimuli, there is still the challenge of visual attention. Visual attention can be roughly thought of as a way of prioritizing a visual field based on interest. Improper allocation of visual attention can mean the difference between catching dinner or going home hungry.

In this research we study the coevolution of prey appearance and visual attention in predators. Prey appearance is evolved via genetic programming, such as in (Sims, 1991a; Reynolds, 2011). Predator vision is evolved using a neurophysiologically-based model of visual attention (Itti

and Koch, 2001). We focus on the effects of environmental complexity on this coevolutionary interaction.

Merilaita (2003) shows that greater background complexity can increase prey detection times by predators. In even earlier work on background matching in camouflage, it was suggested that visual complexity may favor color polymorphism, because there will often be many polymorphisms that can achieve similar patterns (Endler, 1984). These and other works focus on camouflage via background matching, but there are alternative forms of camouflage, in particular, disruptive colorations. In contrast to background matching, where an entity attempts to blend in with the background, disruptive and distractive colorations are patterns that attempt to draw the observer away from the pattern. Disruptive colorations have been shown to be an effective tool for camouflage when tested against live predators (Schaefer and Stobbe, 2006; Cuthill et al., 2005). A number of visual properties significant to predator-prey interactions have been identified in these contexts, including background complexity, prey contrast, and object density (Dimitrova and Merilaita, 2010, 2012, 2014). In our model, the capacity for complexity in prey is greatly enhanced by utilizing a developmental mechanism to produce color images of moderate dimensions, as compared to previous work which explores the selective favorability of simple patterns (Dimitrova and Merilaita, 2014) or directly-represented greyscale images (Bond and Kamil, 2002).

In previous work, we have explored the coevolution of predators and prey (Ficici and Pollack, 1996), finding that such systems are often subject to pathologies such as convergence to mediocre stable states (Ficici and Pollack, 1998), loss of gradient, incorrect focusing, and relativism (Watson and Pollack, 1996). Within the ecology literature, predator-prey systems are commonly studied, including analytical and computational models as well as empirical studies. However, analytical models of predator-prey systems can quickly become infeasible for study as the number of species increases. Furthermore, analysis of phenomena such as predator preference runs into difficulty when accounting for alternative food sources under ecological dynamics (van

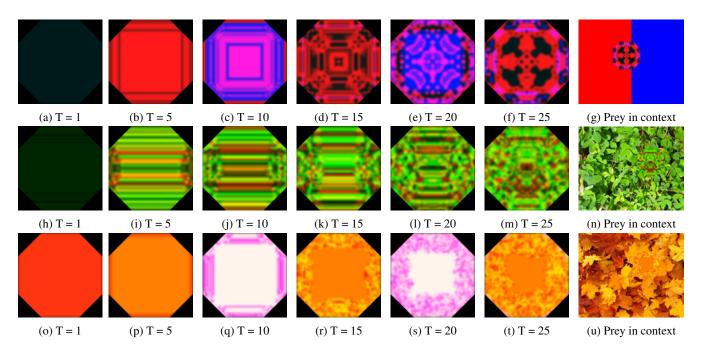


Figure 1: Time-series of prey morphogenesis.

Baalen et al., 2001; Leeuwen and Brännström, 2013).

The pathologies of competitive coevolution can be readily explained in the language of predator-prey systems. Loss of gradient, when one population becomes worse such that it no longer exherts selective pressure on the competing population, or better such that the competing population can no longer maintain engagement. Examples of this would include evolutionary advances in a predator-prey ecosystem where prey can achieve greater escape velocities, leading to diminished returns for predators, and in real ecosystems, probably triggering the predators to seek another food source. Incorrect focusing is when members of one population overspecialize in interactions with specific competitors such that they fail to generalize to other competitive interactions and are prone to extinction. Examples of incorrect focusing are not as common in real ecosystems, where a large number of coevolutionary interactions continually apply selective pressure. In competitive coevolutionary interactions, the quality of an individual is a function of its competitors. This relativism in scoring means that an individual that appears to be more fit to an external observer, may be just as fit as a lower quality individual relative to a given set of competitors. In Nature, this is less of an issue for similar reasons to incorrect focusing, there are continual pressures from many aspects of the ecosystem and environment such that these disambiguations will occur infrequently.

Predator-Prey Coevolution

When simulating predator-prey coevolution, we use a 2-population competitive coevolution model. The prey pop-

ulation consists of a set of genetic programs that encode a generative function for their visual appearance. The predator population consists of a set of numerical weights for a saliency detection algorithm. Prey receive points for not being detected, or causing the predator to incorrectly classify the background as prey. Predators receive points based upon the accuracy of how they perceive the environment.

There are a number of ways to compute the fitness in coevolving populations. We focus on pairwise competitions. Sims uses best v. best competitive coevolution (Sims, 1994), where each population competes against the best individual of the competing population. However, best v. best can lead to incorrect focusing and disengagement by assigning greater fitness to individuals that overspecialize in defeating the champion competitor. Tournaments can reduce the number of comparisons from $O(N^2)$ to O(NlogN), but still represent approximations of a full pairwise competition (Angeline and Pollack, 1993). All v. all competitive coevolution reduces the tendency for focusing, but comes at great computational cost. Nevertheless, we compute the complete payoff matrix via all v. all pairwise competitions to facilitate coevolutionary engagement between all species.

Prey

Prey patterns are produced through a process of algorithmic morphogenesis. The process is much like the standard notions of chemical morphogenesis (Turing, 1952), where a system of reactions determines chemical kinetics while a diffusion system transports chemical species, contributing to pattern formation. However, instead of a standard system of reactions, we employ genetic programming to serve as an

algorithmic chemistry.

The use of genetic programming to evolve images and dynamical systems has been a part of the ALife community since the early years (Sims, 1991b, 1992). These ideas were extended to the evolution of self-constructing and selfrepairing patterns (Miller, 2004). However, much of the work on evolved computer graphics has focused on interactive evolution, perhaps in part because of the complexity of developing a computer vision system capable of scoring images in a meaningful way. The interactive evolution of generative images has recently achieved widespread popularity with the Picbreeder website (Secretan et al., 2011). Of particular relevance to this study is the recent work on the interactive evolution of camouflage (Reynolds, 2011), where a sophisticated texture rendering system is employed to generate patterns for human-guided selection. Finally, alternative biologically-inspired generative representations may be of interest for achieving patterns that may be closer to those of natural systems (Cussat-Blanc and Pollack, 2012).

Genetic programming is an evolutionary method for discovering computer programs (Koza, 1992), where the programs may represent robot controllers, machine learning classifiers, developmental processes, or many other things. The algorithms used to evolve genetic programs are often very similar to those employed by genetic algorithms, with particular exception to how variation is performed. We use the array method of program representation (Koza, 1994) which is most conveniently bounded by program size limits in units of number of nodes (200 nodes for the prey).

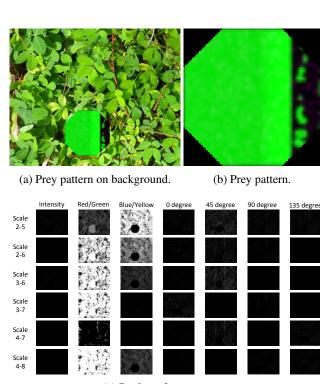
Prey programs are functions of 3 inputs (x, color, and current value) that return a floating point, which are iterated over all color channels at each (x,y) coordinate of the prey, in this case an octagon of radius 41 and (x,y) coordinates are scaled to [-1,1]. This function output is then squished with a hyperbolic tangent. Prey programs can be composed of the following terminal and function set:

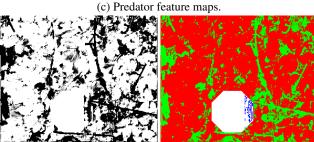
x,color,value,+, -, *, /, iffte, sin, cos, tanh, min, max, abs, hypot, sec, csc, cot, tan, gamma.

Prey are iterated function systems. During each iteration, first the GP program is evaluated once per location and color channel, then a Gaussian filter is convolved with the image as a heuristic pseudo-diffusion. This pseudo-diffusion can introduce artifacts at the boundaries, which we partially alleviate by cropping the prey by a single pixel along the boundary. An example figure can be seen in Figure 1.

Predators

Predator vision is modeled based upon a neurophysiological model of visual attention (Itti and Koch, 2001). Images are broken down into a set of feature maps of intensity, color difference, and orientation, then combined into 42 features taken as differences across multiple scales. Predator genotypes encode weights for each feature map, which are lin-





(d) Predator saliency map.

(e) Predator classification.

Figure 2: Diagram of predator's visual system. 2a shows the prey pattern situated on the grassy background. 2b shows a zoom-in of the prey pattern. 2c shows the weighted feature maps used by an example predator. 2d shows the corresponding saliency map for the feature maps in 2c. 2e shows the same example predator's classification of 2a, where red indicates incorrectly classified background, green indicates correctly classified background, white indicates correctly classified prey, and blue indicates incorrectly classified prey.

early combined into a saliency map. The saliency map encodes the priority of attention at each location in the visual field. We allow the saliency map to take on both positive and negative values, where positive values indicate a prediction of the prey's position. For a more detailed description of the components of the vision model, see (Itti and Koch, 2001).

Saliency-based models of visual attention have a long history in studies of the neuroscience of vision (Niebur et al., 1993; Itti and Koch, 2001; Borji and Itti, 2013). There are now many algorithms for visual attention, some based on

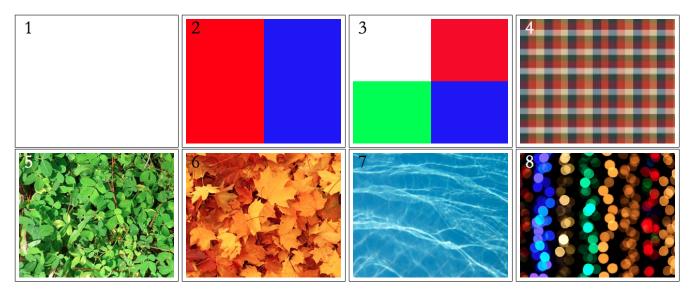


Figure 3: Prey environments used in experiments.

neurophysiology and some engineered for optimality. A comprehensive review of current visual attention algorithms can be found in (Borji and Itti, 2013). We use a bottom-up model based upon (Itti and Koch, 2001), which has been shown to correlate with human eye movements (Parkhurst et al., 2002; Itti, 2005). In the traditional model of Itti, where feature maps are aggregated hierarchically into the saliency map, first by grouping features by type into conspicuity maps, then combining conspicuity maps to compute the saliency map. Our model uses the simplification of combining all feature maps to immediately compute the saliency map.

Evolutionary Algorithm

We use a genetic algorithm for both the predator and prey populations. Individuals are selected with tournament selection using tournament sizes of 3. Selected individuals are mutated (45%), crossed over (45%), or replicated (10%) in the successive generation. Prey are crossed over using standard subtree mutation and crossover (Koza, 1992), while predators are mutated by adding a vector of small Gaussian mutations to the genome and recombined via uniform crossover. The evolutionary algorithm is run for 500 generations in all experiments in this study.

Experiments

We consider a number of environmental backgrounds of varying complexity. We obtained 5 naturally patterned images¹ and created 3 simple images, shown in Figure 3. Images were prepared by cropping regions to a size of 300x240. While a number of methods have been developed

within the computer vision community to characterize image complexity, we report on the JPEG file size. The concept of complexity resides at the heart of the field of data compression, hence our choice to use it as a metric of image complexity. Image sizes in the same order as Figure 3: 446B, 1.3kB, 1.7kB, 14k, 47k, 123k, 93k, and 126k. During simulations prey position and rotation is randomly determined. Prey position is randomly chosen such that the prey resides entirely within the background environment, and rotation is uniformly chosen from all 4 possible 90 degree rotations. 25 independent trials are conducted for each background environment². Due to the computational costs of simulating prey morphogenesis, predator vision, and computing complete pairwise payoff matrices, we use population sizes of 100 for both the predators and prey. The same random seeds are used for each background, such that the initial populations have the same constituents for each random seed, but then quickly diverge as selection and mutation vary the populations. In Figures 4, 5, 6, and 7 we use two colors, red and blue, to indicate data reported for the population average and best individual, respectively.

When measuring the degree of background matching that is present in a given population over evolutionary time, we use the same 42-D feature vector that is used by predator vision. Merilaita and Lind (2005) previously suggested that quantification of background matching in prey can be misleading if it isn't computed with respect to the predator's perception. When measuring the distance between a prey and the background we take the Euclidean distance between the average feature vector of the background image and the

¹Images are public-domain, and are retrieved from http://www.publicdomainpictures.net/.

²Due to computational difficulties, a few runs were incomplete. Complete runs per background (ordered as in Figure 3): 24, 21, 25, 23, 25, 24, 22, 23

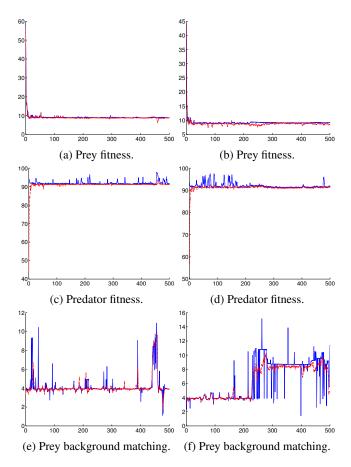


Figure 4: Example evolutionary trajectories of fitness and background similarity. X-axis is time. Subcaption indicates respective Y-axis.

average feature vector of the prey. For evolving populations we measure the average of the average feature vector of all prey in the population.

Background Matching

Background matching is generally successful for all backgrounds in this study (see Figures 1n and 1u for particularly compelling examples), with a noteable exception and some unexpected insights. First, let us reiterate the argument initially proposed by Endler (1984). The regions of the background on which the prey are selected lead to correlations between the prey patterns and those regions of the background, because better matches generally win. In our study the random repositioning and rotation of the prey means that prey effectively have a uniform probability of being tested at each location in the image. Therefore, prey that match the average background region in the image are predicted to generally be a better fit. However, a priori it is unclear what prey patterns will appear in environments with stark differences, or natural complexity. In Figure 1g, we show one example of such a situation, where on a background of

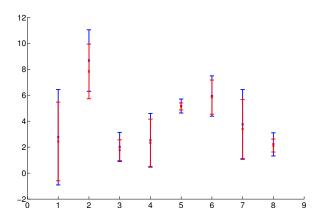


Figure 5: Background matching in prey. Y-axis indicates distance from the background, see Experiments for description of distance from background. Smaller means a better match. X-axis is the ordering of backgrounds in Figure 3.

half blue and half red the prey adopts a strategy that contains both red and blue which can be easily mutated to favor one color over the other. This is a common trend in our results; prey evolved on backgrounds with large differences at the scale of the prey favor polymorphic populations.

Interestingly, there is not a clear relationship between prey fitness and prey background matching. While this is not surprising in some senses, because fitness under competitive coevolution is a function of the competing population, where an improvement in one population masks an improvement in the competing population. It does suggest that the presence of a third-party in competitive coevolution, the environment, can non-trivially alter the coevolutionary dynamics. We discuss this in greater detail when reflecting on the representation of predator vision and prey appearance. Nevertheless, because we randomly position and orient prey, we suggest that it may be worthwhile to pursue non-random prey movement to reduce the background-averaging tendency of prey.

Coevolutionary Dynamics

On first glance, the coevolutionary dynamics in this study are fraught with the Red Queen effect. Originally presented as a dynamic describing the constant probability of extinction (van Valen, 1973), the Red Queen effect is often described as the requirement that "takes all the running you can do, to keep in the same place" (Carroll, 1871). Figure 4 shows 2 example evolutionary trajectories from the white background. In one (Figures 4a, 4c, and 4e), the prey population evolves to a near perfect solid white pattern (see the 8th image in Figure 8), and in the other (Figures 4b, 4d, and 4f), the prey population evolves first to a light grey then fixates on a yellow cross pattern for an extended period (see the 17th image in Figure 8) and ends at generation 500 with a black prey pattern as the champion. Nevertheless, by simply

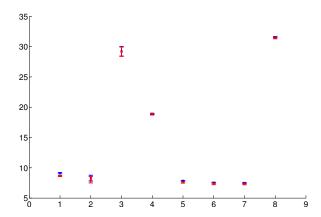


Figure 6: Prey fitness. X-axis is time. Y-axis is prey fitness. Bigger is better.

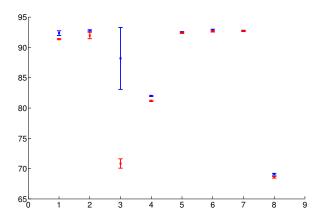


Figure 7: Predator fitness. X-axis is time. Y-axis is predator fitness. Bigger is better.

inspecting the fitness of the predators and prey it is not obvious that there has been any major change in the predatorprey interactions, certainly not to the degree that has previously been described as loss of gradient, suggesting that this dynamic is akin to relativism (Watson and Pollack, 2001).

From the set of backgrounds we used in these experiments, simple backgrounds led to the greatest degree of relativism. In particular, multiple trials with the white background resulted in evolutionary trajectories that first converged to near-white prey patterns, followed by a divergence from background similarity leading to colored patterns. We suspect that one of the reasons for this relativism was related to background discrimination ability in predators. In particular, predators make absolute decisions regarding the location of prey, where a slight change in prey hue can lead to predator misclassification. Probabilistic decisions may facilitate environmental engagement by allowing predators some degree of uncertainty in detecting prey.

Representation

Representation often has a significant impact on evolutionary dynamics by, amongst other effects, affecting the genetic distances between phenotypes and constraining the space of possible phenotypes. We utilize a genetic programming representation for prey which, by incorporating a heuristic pseudo-diffusion, is capable of producing a wide array of prey patterns. Yet these patterns are likely to have limitations that arise from both the function set used as the basis for prey programs and the instabilities that arise under diffusion (although in some systems diffusive instabilities can provide macroscopic stability, such as Turing patterns (Turing, 1952)). Furthermore, there is no interaction between prey patterns and the environment. Not only do some species exhibit active camouflage, but there are links between visible pigmentation and diet (Whitehead et al., 2012).

We suspect that the representation of predator vision is one of the more significant variables worthy of future investigation. Figure 7 suggests that predators have difficulty with multi-colored images. Prey also perform better on backgrounds with homogeneous color schemes, both in terms of fitness and background similarity, Figures 6 and 5 respectively. Due to the competitive nature of the model, it is unclear whether the bias towards homogeneous color schemes is due to the predators or the prey, but we suspect that it is due to the representation of predator vision. By considering alternative visual attention algorithms, such as those reviewed in (Borji and Itti, 2013), we expect to observe prey patterns that reflect the properties of the visual attention algorithm.

Conclusions

We have presented a model of the coevolution of predator vision and prey camouflage, where prey utilize a developmental process to form complex multicolored patterns and predators use a neurophysiologically-based model of visual attention. Prey successfully evolve to match their background, with some exceptions that are predicted to stem from randomized prey movement, and predators successfully evolve the ability to discriminate between prey and background. We observe a type of coevolutionary relativism, where competing populations remain engaged while drifting away from their environment. This type of divergence from background matching generally happens in simpler backgrounds, leading to the hypothesis that background complexity can facilitate coevolutionary engagement on background matching problems.

While other coevolutionary studies have found that pursuer-evader tasks can lead to mediocre stable states (Ficici and Pollack, 1998), we find that coevolution in our model is generally engaged and leads to effective background matching. We suggest that it is the richness of the model that facilitates engagement. However, background

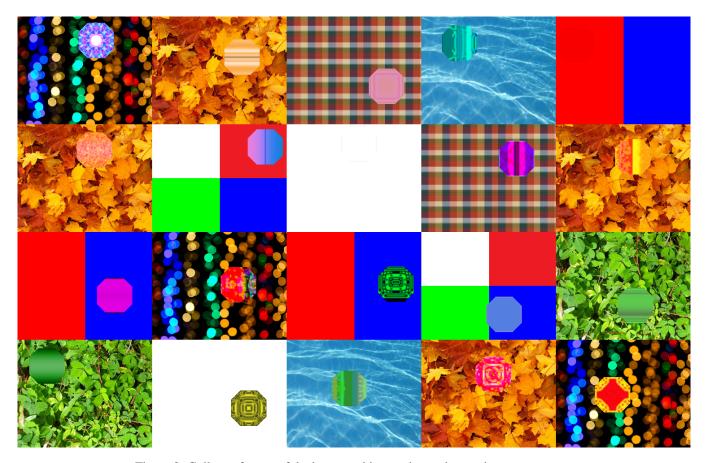


Figure 8: Collage of some of the best matching, and most interesting prey patterns

matching is not always achieved. In particular, simple environments with stark differences where only partial background matching is possible tend to favor prey that can readily mutate to match regions of the environment. This suggests that polymorphic populations may be an effective response to a highly variable background. In the case of complex backgrounds we find that predators and prey are generally more engaged, and background complexity appears to support the advance of the coevolutionary ratchet, leading to effectively camouflaged prey.

Future Work

There are many avenues for future research stemming from this work. We suggest only two examples. First, many natural predators learn over time, whereby they may become better at identifying prey during their lifetime, see (Troscianko et al., 2013). Second, in Nature, prey patterns act are signals, both to prey and to mates. It may prove interesting to consider the dynamics of mate signaling which has led to such brilliant patterns as peacocks' plumage. Along this line, we have previously discovered models and corresponding parameters capable of leading to the emergence of such costly signaling (Harrington et al., 2012).

Acknowledgements

KIH is funded by the Brandeis University School of Computer Science. Computational support was provided by the Brandeis HPC.

References

Abbott, K. (2010). Background evolution in camouflage systems: A predatorprey/pollinator-flower game. *Journal of theoretical biology*, 262(4):662–678.

Angeline, P. J. and Pollack, J. B. (1993). Competitive Environments Evolve Better Solutions for Complex Tasks. In Forrest, S., editor, *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, pages 264–270, University of Illinois at Urbana-Champaign. Morgan Kaufmann.

Bond, A. and Kamil, A. (2002). Visual predators select for crypticity and polymorphism in virtual prey. *Nature*, 415(6872):609–613.

Borji, A. and Itti, L. (2013). State-of-the-art in visual attention modeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):185–207.

Carroll, L. (1871). Through The Looking-Glass.

Cussat-Blanc, S. and Pollack, J. (2012). Using Pictures to Visualize the Complexity of Gene Regulatory Networks. In *Artificial Life*, pages 491–498.

- Cuthill, I., Stevens, M., and Sheppard, J. (2005). Disruptive coloration and background pattern matching. *Nature*, 434(7029):72–74.
- Dimitrova, M. and Merilaita, S. (2010). Prey concealment: visual background complexity and prey contrast distribution. *Behavioral Ecology*, 21(1):176–181.
- Dimitrova, M. and Merilaita, S. (2012). Prey pattern regularity and background complexity affect detectability of background-matching prey. *Behavioral Ecology*, 23(2):384–390.
- Dimitrova, M. and Merilaita, S. (2014). Hide and seek: properties of prey and background patterns affect prey detection by blue tits. *Behavioral Ecology*, page art130v1.
- Endler, J. (1984). Progressive background in moths, and a quantitative measure of crypsis. *Biological Journal of the Linnean Society*, 22(3):187–231.
- Ficici, S. G. and Pollack, J. B. (1996). Coevolving Communicative Behavior in a Linear Pursuer-Evader Game. In *Proceedings* of the Fifth International Conference of the Society for Adaptive Behavior, pages 505–514.
- Ficici, S. G. and Pollack, J. B. (1998). Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In *Proceedings of the sixth international conference on Artificial life*, pages 238–247.
- Harrington, K. I., Ozisik, A. P., and Pollack, J. (2012). The Effects of Finite Populations and Selection on the Emergence of Signaling. In *Proceedings of Artificial Life XIII*, pages 194–201.
- Itti, L. (2005). Quantifying the contribution of low-level saliency to human eye movements in dynamic scenes. *Visual Cognition*, 12(6):1093–1123.
- Itti, L. and Koch, C. (2001). Computational modelling of visual attention. *Nature Reviews Neuroscience*, 2(3):194–203.
- Koza, J. (1992). Genetic programming: on the programming of computers by means of natural selection.
- Koza, J. R. (1994). Genetic programming II: automatic discovery of reusable programs, volume 1.
- Leeuwen, E. V. and Brännström, A. (2013). A generalized functional response for predators that switch between multiple prey species. *Journal of theoretical biology*, 328:89–98.
- Merilaita, S. (2003). Visual background complexity facilitates the evolution of camouflage. *Evolution*, 57(6):1248–1254.
- Merilaita, S. and Lind, J. (2005). Background-matching and disruptive coloration, and the evolution of cryptic coloration. Proceedings of the Royal Society B: Biological Sciences, 272(1563):665–670.
- Miller, J. (2004). Evolving a self-repairing, self-regulating, French flag organism. In *Genetic and Evolutionary ComputationGECCO* 2004.
- Morin, S., Shepherd, R., and Kwok, S. (2012). Camouflage and display for soft machines. *Science*, 337(6096):828–832.
- Niebur, E., Koch, C., and Rosin, C. (1993). An oscillation-based model for the neuronal basis of attention. *Vision Research*, 33(18):2789–2802.

- Parkhurst, D., Law, K., and Niebur, E. (2002). Modeling the role of salience in the allocation of overt visual attention. *Vision research*, 42(1):107–123.
- Reynolds, C. (2011). Interactive evolution of camouflage. *Artificial Life*, 17(2):123–136.
- Schaefer, H. and Stobbe, N. (2006). Disruptive coloration provides camouflage independent of background matching. *Proceedings of the Royal Society B: Biological Sciences*, 273(1600):2427–2432.
- Secretan, J., Beato, N., D'Ambrosio, D. B., Rodriguez, A., Campbell, A., Folsom-Kovarik, J. T., and Stanley, K. O. (2011). Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary Computation*, 19(3):373–403.
- Sims, K. (1991a). Artificial Evolution for Computer Graphics. *Computer Graphics*, 25(4):319–328.
- Sims, K. (1991b). Artificial evolution for computer graphics.
- Sims, K. (1992). Interactive evolution of dynamical systems. In Varela, F. J. and Bourgine, P., editors, *Toward a Practice* of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, pages 171–178, Paris, France. MIT Press.
- Sims, K. (1994). Evolving virtual creatures. In *Proceedings of the* 21st annual conference on Computer graphics and interactive techniques, number July, pages 15–22. ACM.
- Troscianko, J., Lown, A., Hughes, A., and Stevens, M. (2013). Defeating crypsis: detection and learning of camouflage strategies. *PloS one*, 8(9):e73733.
- Turing, A. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 237(641):37–72.
- van Baalen, M., Kivan, V., van Rijn, P. C., and Sabelis, M. W. (2001). Alternative food, switching predators, and the persistence of predatorprey systems. *The American Naturalist*, 157(5):512–524.
- van Valen, L. (1973). A new evolutionary law. *Evolutionary Theory*, 1:1–30.
- Watson, R. and Pollack, J. (2001). Coevolutionary dynamics in a minimal substrate. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-01*, pages 702–709. Citeseer.
- Watson, R. A. and Pollack, J. B. (1996). Coevolutionary Dynamics in a Minimal Substrate. In *Proceedings of the Genetic and Evolutionary Computation Conference*, number 1992, pages 702–709.
- Whitehead, R., Re, D., Xiao, D., Ozakinci, G., and Perrett, D. (2012). You are what you eat: Within-subject increases in fruit and vegetable consumption confer beneficial skin-color changes. *PloS one*, 7(3):e32988.

What happened to my genes? Insights on gene family dynamics from digital genetics experiments

C. Knibbe^{1,2} and D. P. Parsons¹

¹INRIA Rhône-Alpes, Montbonnot F-38322, France

²Université de Lyon, Université Lyon 1, CNRS, UMR5205, LIRIS, Villeurbanne, F-69622, France carole.knibbe@univ-lyon1.fr

Abstract

Gene families are sets of homologous genes formed by duplications of a single original gene. Inferring their history in terms of gene duplications, gene losses and gene mutations yields fundamental insights into the molecular basis of evolution. However, phylogenetic inference of gene family evolution faces two difficulties: (i) the delimitation of gene families based on sequence similarity, and (ii) the fact that the models of evolution used for reconstruction are tested against simulated data that are produced by the model itself. Here, we show that digital genetics, or in silico experimental evolution, can provide thought-provoking synthetic gene family data, robust to rearrangements in gene sequences and, most importantly, not biased by where and how we think natural selection should act. Using aevol, a digital genetics model with an abstract phenotype but a realistic genome structure, we analyzed the evolution of 3,512 synthetic gene families under directional selection. The turnover of gene families in evolutionary runs was such that only 21% of those families would be accessible for classical phylogenetic inference. Extinct families showed patterns different from the final, observable ones, both in terms of dynamics of gene gains and losses and in terms of gene sequence evolution. This study also reveals that gene sequence evolution, and thus evolutionary innovation, occurred not only through local mutations, but also through chromosomal rearrangements that re-assembled parts of existing genes.

Introduction

How do new genes arise? Do they evolve mainly by local mutations or domain shuffling? Which events drive them to extinction? These questions are fundamental to understand the evolutionary dynamics of living systems. Because the preservation of soft tissues is rare in fossil records, paleontology provides precious but limited knowledge of the past of the living world. The study of molecular evolution thus largely relies on the analysis of extant genes, which may or may not be a representative sample of genetic diversity throughout the evolution of life. Central in this analysis is the notion of gene family, defined as a set of homologous genes formed by duplications of a single original gene. Insights into the evolutionary dynamics at the molecular level are obtained by inferring the evolutionary history of gene gains and losses and of gene mutations in a gene family.

The usual strategy to identify gene families consists in detecting significant sequence similarities in gene or protein sequences. This method is inherently biased towards the detection of families that evolve mainly through local mutations rather than domain shuffling. As Song et al. (2008) make it clear, "multidomain sequences, especially those with promiscuous domains that occur in many contexts, are frequently excluded from genomic analyses due to the lack of a theoretical framework and practical methods for detecting multidomain homologs". Efforts are thus ongoing to develop multidomain homology identification methods (Geer et al., 2002; Enright et al., 2002; Lin et al., 2006; Song et al., 2008; Jachiet et al., 2013).

Once gene families have been identified, their evolutionary histories are inferred, using implicit or explicit models of evolution to describe the patterns of DNA base substitution and amino acid replacement (Liò and Goldman, 1998) and the patterns of gene gains and losses (Arvestad et al., 2004; Vilella et al., 2008; Akerborg et al., 2009; Rasmussen and Kellis, 2012; Boussau et al., 2013). These models of evolution make assumptions – for example, the model used by Vilella et al. (2008) assumes that gene duplications and deletions are rare events, and that duplication followed by complementary gene losses on the left and right branches of a duplication node is an unlikely scenario. Most models also assume that different gene families evolve independently, while a single duplication or deletion can actually span several genes. The scarcity of well-preserved ancient DNA samples makes it difficult to really test these hypotheses. The common practice to test a phylogenetic method is thus to simulate artificial sequences to generate benchmarks. However, these artificial sequences are usually generated with the same general model of evolution as the one used by the phylogenetic method being tested, with only minor differences (see for example Rasmussen and Kellis (2012); Boussau et al. (2013)). There is thus a form of circularity in the overall process, which could leave some important aspects of evolutionary dynamics in the dark.

To provide better benchmarks for phylogenetic inference, some simulators like EvolSimulator (Beiko and Charlebois,

2007) and ALF (Dalquen et al., 2012) have been developed independently of a particular phylogenetic inference method. For example, ALF uses classical models of evolution at the gene sequence level, but allows for the duplication or loss of several consecutive genes at once. However, both ALF and EvolSimulator simulate only one sequence per species. The action of natural selection is incorporated in the mutational process, in the sense that only mutations assumed to be neutral or beneficial are simulated. For example, mutations that would lead to the formation of a stop codon (nonsense mutations) are not allowed in ALF (Dalquen et al., 2012). In EvolSimulator, specific probabilities of duplication and loss are pre-assigned to each gene (Beiko and Charlebois, 2007). Simulators of sequence evolution are also being developed in population genetics (reviewed by Hoban et al. (2012)), to predict the molecular polymorphism expected under various demographic scenarios. All individuals of the population are simulated but the genomic architecture is usually fixed, implying that gene gains or losses are not allowed. Deleterious events are allowed but the distribution of fitness effects is predefined at each locus.

Experimental evolution of microbes is a much more direct way to study evolution. Although the time scale of these laboratory experiments is short compared to those at stakes in phylogenetic reconstruction, gene gains and losses do occur (Nilsson et al., 2005; Blount et al., 2012; Tenaillon et al., 2012; Maharjan et al., 2013; Payen et al., 2014) and frozen samples allow for a partial fossil record. In silico experimental evolution, or digital genetics (Adami, 2006), can bring a complementary perspective by providing fast synthetic genomic data, in which – in contrast to other types of simulators - the action of natural selection on genomic sequences is not predetermined by the user. In digital genetics, an abstract artificial chemistry is used to compute a phenotype from a genotype, and selection is based on the phenotype, not on the genotype. The Avida platform has already been used to test the effect of selection on phylogenetic reconstruction methods (Hagstrom et al., 2004; Hang et al., 2007). Here, we show how a digital genetics platform with a realistic genome structure can be used to directly study gene family evolution, with both local mutations and rearrangements. The exhaustive knowledge of all evolutionary events allows for the identification of gene families even when sequence similarity could be impaired by rearrangements that shuffled parts of coding sequences.

After a presentation of the model, called aevol (http://www.aevol.fr), we study the evolution of ten independent populations under directional selection, yielding 3,512 synthetic gene families. Using a new postprocessing tool designed for this purpose, we analyze the dynamics of genes within the context of those families – how and how often new genes are created, how and how often they are lost, how many and what types of mutational events occur in their

sequences. By going beyond the simple time series of gene number, we show that our usual interpretation of gene number evolution in aevol was partly wrong. Not all new genes arise by duplication-divergence. Many are also created from previously non-coding sequences, after either a local mutation or a rearrangement. We also show that there is a high turnover of gene families, many of them lasting only a few hundreds or thousands of generations. This implies that final extant genes give only a partial insight into the dynamics of gene family evolution. Moreover, our analysis reveals that rearrangements do not restrict themselves to changing gene number and gene order. They also play a significant role in gene sequence evolution, and thus in evolutionary innovation, by rearranging parts of existing genes.

Aevol: A digital genetics model

Aevol is a digital genetics model that simulates the evolution of a population of N haploid organisms through a process of variation and selection. It was designed to study the evolution of genome structure (Knibbe et al., 2007; Beslon et al., 2010; Parsons et al., 2010; Frenoy et al., 2013; Batut et al., 2013). Thus, the design of the model focuses on the realism of the genome level and of the mutational process, while the selection process simply relies on a one-dimensional curvefitting task.

Genome representation

Each artificial organism owns a chromosome whose structure is inspired by prokaryotic genomes. It is organized as a circular double-strand binary string containing a variable number of genes separated by non-coding sequences (figure 1). Genes are delimited by predefined signaling sequences indicating transcription and translation start and stop. Transcription initiates at promoters, defined in the model as sequences that differ from an (arbitrarily chosen) 22-bp consensus sequence by $d \leq 4$ mismatches. When a promoter is found, the transcription proceeds until a terminator is reached. Terminators are defined as sequences that would be able to form a stem-loop structure, as the ρ -independent bacterial terminators do. In the following experiments, terminators had the structure $abcd * * * \overline{d}\overline{c}b\overline{a}$, where $\overline{a} = 0$ if a=1, and conversely. The expression level e of an mRNA is determined according to the similarity of its promoter to the consensus: $e = 1 - \frac{d}{5}$.

Transcribed sequences (mRNAs) do not necessarily contain coding sequences. The translation initiation signal is the motif 011011****000 (Shine-Dalgarno-like sequence followed, a few base-pairs away, by a START codon). When this signal is found on a mRNA, the downstream sequence is read three bases (one codon) at a time until the termination signal, the STOP codon 001, is found on the same reading frame. Each codon lying between the initiation and termination signals is translated into an abstract "amino-acid" using an artificial genetic code (Figure 1).

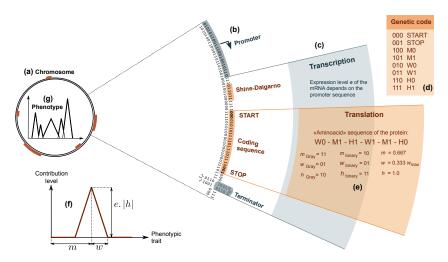


Figure 1: In the model, each organism owns a circular double-strand binary chromosome (a) along which genes are delimited by predefined signal sequences (b). Promoters and terminators mark the boundaries of RNAs (c) within which coding sequences are in turn identified between a Shine-Dalgarno-START signal and an in-frame STOP codon. Each coding sequence is then translated into a protein sequence using a predefined genetic code (d). This protein sequence is decoded as three real parameters called m, w and h (e). Proteins, phenotypes and environments are represented similarly through mathematical functions that associate a level to each abstract phenotypic trait in [0,1]. The contribution of a protein is a piecewise-linear function with a triangular shape, with position m, half-width w and height h (f). All proteins encoded in the chromosome are then combined to compute the phenotype (g), which is compared to the environmental target to compute the fitness of the individual.

Protein function and phenotype computation

In the model, we assume that there is an abstract, continuous one-dimensional space $\Omega = [0, 1]$ of phenotypic traits. Each protein contributes positively or negatively to a subset of phenotypic traits, and is modeled as a mathematical function that associates a contribution level between -1.0 and 1.0 to each phenotypic trait. For simplicity, we use piecewiselinear functions with a symmetric, triangular shape (figure 1). In this way, only three numbers are needed to characterize the contribution of a protein: The position m ($m \in \Omega$) of the triangle on the axis, its half-width w and its height h (positive or negative). The protein thus contributes to the phenotypic traits in [m-w, m+w], with a maximal contribution for the traits closest to m. Thus, various types of proteins can co-exist, from highly efficient and highly specialized ones (low w, high h) to polyvalent but poorly efficient ones (high w, low h).

In this framework, the sequence of each protein is decomposed into three interlaced binary subsequences that will in turn be decoded as the values for the $m,\,w$ and h parameters. For instance, the codon 010 (resp. 011) is translated into the single amino acid W0 (resp. W1), which means that it adds a bit 0 (resp. 1) to the Gray code of w. (The Gray code is a variant of the traditional binary code. It is widely used in evolutionary computation because it avoids the so-called Hamming cliffs: in the Gray code representation, consecutive integers are assigned bit strings that differ by only one bit.). Small mutations in the coding sequence

(point mutations, indels, possibly causing frame shifts) can change these parameters and hence change the contribution of the protein to the phenotypic traits.

Once all the proteins encoded on the genotype of the organism have been identified, their contributions are combined to get the final level for each phenotypic trait. This is done by summing the mathematical functions of all proteins and keeping the result bounded between 0 and 1.0. The resulting piecewise-linear function $f_P:\Omega\to[0,1.0]$ is called the phenotype of the organism. It indicates the level of each phenotypic trait in Ω .

Environment, adaptation and selection

In the model, fitness depends on the difference between the levels of the phenotypic traits, and target levels defined by a mathematical function $f_T:\Omega\to[0,1.0]$. This target function indicates the optimal level of each phenotypic trait in Ω and is called the environmental target, or target for short. Here, f_T was made up of three gaussian lobes with standard deviation 0.05 and maximal height 0.5, centered on x=0.2, 0.6 and 0.8 respectively. It was kept constant over evolutionary time. Adaptation was specifically measured by the gap $g=\int_{\Omega}|f_T(x)-f_P(x)|dx$ between f_P and f_T . The lower the gap, the fitter the individual. This measure penalizes both the under-realization and the over-realization of each phenotypic trait.

In the current version of Aevol, the population size is constant (here N=1,000 individuals) and the population is

entirely renewed at each generation. A probability of reproduction is assigned to each individual according to its gap and a multinomial drawing determines the actual number of offsprings each individual will have. Here, we used the so-called "fitness-proportionate" selection scheme, where the probability of reproduction of an individual with gap g was $\frac{e^{-kg}}{\sum_{i=1}^N e^{-kg_i}}.$ Here the environment was considered perfectly mixed, but a spatial grid structure where an individual competes only with its neighbors can also be used.

Mutations and rearrangements

During their replication, genomes can undergo local mutations (point mutations and small insertions or deletions of 1 to 6 bp) and chromosomal rearrangements (duplications, deletions, translocations and inversions). The breakpoints for these rearrangements are randomly chosen on the chromosome. A translocation is here defined as moving a segment to another position on the chromosome. Other versions of the platform exist that allow for plasmids, in which case translocations can move subsequences from the chromosome to the plasmids and conversely. This feature was not used here. Similarly, although lateral transfer is possible in aevol, we did not use it in these experiments, to keep the setup simple in this first study of gene family evolution. The rates of the different types of genetic modification occurs are defined per-base, per-replication.

Workflow with the software suite

The typical workflow with the aevol software suite starts with the preparation of the initial population following the initialization method chosen by the user (with a random genome or with a mix of already evolved ones, for a competition assay for example). The second step is the evolutionary run itself. Depending on the parameters, a run of 100,000 generations may take from several hours up to several days. The population size is important, but the spontaneous rates of rearrangements as well, since they influence the evolved genome size (Knibbe et al., 2007). If ancestry relationships and mutations have been recorded during the run, and if individuals were asexual, it is possible, as a third step, to extract the line of descent of the best final individual from the recorded data and to replay the evolutionary events that occurred along this successful lineage. Except for the very last mutations that were possibly still segregating in the population, the replayed mutations are those that were fixed.

Results

We let 10 populations of 1,000 haploid asexual individuals evolve independently under directional selection during 100,000 generations. The spontaneous rate of each type of mutational event was set to 10^{-5} per bp. At the beginning of a run, all organisms of the population were initialized with a same random sequence of 5,000 bp containing at least one

gene. This initial sequence was different for each population. In practice, populations started with either one or two genes. With this setup, we do not aim at mimicking the origin of life but rather the adaptation to a novel niche. Indeed, in the model, an individual without any gene on its chromosome can still replicate itself and express its genes: "Core genes" for replication, transcription, translation are assumed to be implicitly present in each individual and their evolution is not modeled. What is actually simulated is the evolution of the non-essential subset of the genome, when the population faces a new environment.

As shown by Figure 2, genome evolution on the successful lineages starts with a phase of expansion, where new genes are massively acquired, along with much non coding DNA. This excess DNA is then progressively removed from the genome, while gene acquisition slows down. This pattern was already observed in (Knibbe et al., 2007), in a similar setup. Here, we went deeper into the analysis of gene repertoire dynamics by tracking the fate of each gene, as well as the paralogy relationships between genes.

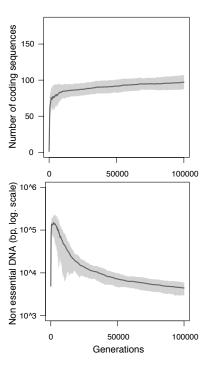


Figure 2: Evolution of genome size on the line of descent of the final best individuals. The shaded area indicates the standard deviation across repetitions. Non essential DNA is defined as DNA that can be removed without changing the phenotype. It includes intergenic DNA, but also the transcribed but untranslated regions (UTRs).

For each repetition, evolutionary events on the line of descent of the best final individual were replayed. Each gene in the initial genome was tagged and considered the root of a gene family, which was stored as a binary tree. When replay-

ing the mutational events, the fate of each gene was followed and recorded mutation after mutation. We considered in this analysis that a gene was composed of its coding sequence and of its "upstream region", defined as the sequence located between the first base pair of the first promoter of the coding sequence and the first base pair of the Shine-Dalgarno-start signal for translation initiation.

Figure 3 shows an example of a small gene family. The topology of the tree indicates the dynamics of gene duplications and losses. Because the exact timing of events is known, the branch lengths represent the real time elapsed, in number of generations. The branches are annotated with the mutational events that affected either the coding sequences or their upstream regions. These events can either be local mutations or chromosomal rearrangements. Indeed, aevol, along with the "ARN" model (Banzhaf, 2003) and potentially the model of early metabolism by Ullrich et al. (2011), is one of the few digital genetics models in which the breakpoints of chromosomal rearrangements are not constrained to intergenic regions. They operate at the sequence level rather than at the gene level and are thus blind to the genic/intergenic status of the sequences they disrupt. As a consequence, they can modify gene content and gene order, but also generate variability in gene sequences.

Number of gene families

In previous studies with aevol, which mostly relied on the time series of gene number like on Figure 2, we thought that most evolved genes ultimately descended from the initial one(s). Indeed, because simulations started with only one or two genes and because several initiation signals are necessary for a sequence to be coding, one would expect that most genes would be created by a duplication-divergence process and that each run would thus contain one or two gene families only. There were actually on average 351.2 ± 158.3 gene families per evolutionary run, contrary to what we expected. This is not an artifact due to a saturation of the phylogenetic signal, because gene family identification is here based on the exact knowledge of all events and not on sequence similarity, and is thus insensitive to such a saturation. Thus, de novo gene creation was not rare. On average, in a run, the gene families of the initialization represented only 0.4% of all families and 10.8% of the final genes. This fits with a recent analysis of proto-genes candidates in the genome of the yeast S. cerevisiae, which suggested that "de novo gene birth may be more prevalent than sporadic gene duplication" (Carvunis et al., 2012). In our synthetic dataset, 55% of de novo gene creations were due to a local mutation and around 44% were due to a chromosomal rearrangement.

The large variation across runs in the number of families stems from a bimodal distribution, with seven runs centered around 281 families (hereafter called group A) while three other runs (called group B) are centered around 632 families. This variation is due to the initial phase of genome

expansion, since on average 61% of the gene families in these three runs were extinct before generation 10,000. At the end of the runs (t=100,000 generations), on average 73.4 ± 6.2 gene families were still active in each run, meaning that they had at least one remaining gene. Thus, the families that would be accessible for classical phylogenetic inference would represent only 21% of the gene families that played a role in the evolutionary history of the evolved populations.

Size of gene families

What is usually called the family size is the number of non-extinct leaves at the time of observation. Here, the mean family size at t=100,000 was 1.36 ± 0.1 non-extinct leaves, meaning that each family had on average 1.36 sibling genes (paralogs) in the evolved genome. In real genomes, gene family size is known to follow a power-law distribution (Huynen and van Nimwegen, 1998), with a vast majority of very small gene families and a few very large families. As shown by Figure 4, the family sizes obtained here do not span enough orders of magnitude to conclude to a power-law distribution. However, in all runs, the vast majority of families had size 1, while only one or two families had a size larger than 4.

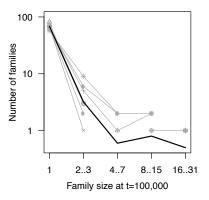


Figure 4: Distribution of gene family size at t=100,000. The different symbols correspond to the different repetitions and the black curve is the mean frequency over the ten repetitions. Following Huynen and van Nimwegen (1998), family sizes were binned exponentially and both axes are logarithmic. Missing symbols correspond to a frequency of 0.

Rates of gene duplication and loss

When taking all families of a run into account, a gene gain by duplication occurred on average every 212 generations in group A, and every 5.5 generations in group B. A gene loss occurred every 158 generations on average in group A, and every 5.4 generations in group B (63% of gene losses happened by the complete deletion of the gene, 16% were due to another chromosomal rearrangement and 21% were due to a local mutation). However, these rates of gene

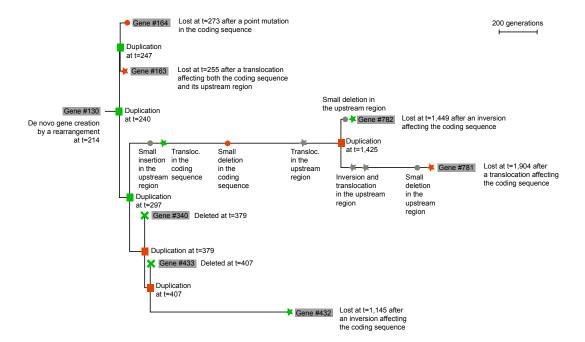


Figure 3: Example of a small gene family. This is the fourth gene family of the third run. It was born at t=214 and went to extinction at t=1,904. Squares indicate duplications, crosses indicate deletions, stars indicate inversions or translocations, and circles indicate point mutations, small insertion or small deletions. Gray events were neutral, while red events were deleterious and green events were beneficial. The topology of the tree was drawn with NJPlot (Perriere and Gouy, 1996).

gains and losses are somehow misleading because 90% of the gene duplications and gene losses occurred before generation 5,500. Hence, during the initial phase of genome expansion, the gene content is extremely dynamic. Besides, duplications and deletions generally encompass several neighboring genes, thereby creating strong correlations across families and inside different subtrees of a family. For example, in the first run, there were 604 gene duplications but they were concentrated on 85 different generations only.

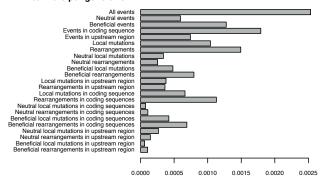
Evolutionary rates of gene sequences

On each branch of each gene family, we counted and classified the events that modified the gene sequence without killing it, and divided these counts by the branch length in number of generations. Those rates per gene per generation were averaged over all branches of all gene families of all runs to produce Figure 5A. It shows that the coding sequences underwent more changes than the upstream region. This an expected result given that (in the model) changes in the coding sequences can change the phenotypic traits to which the gene contributes, whereas changes in the promoter can just modulate the level of a gene contribution. Both local mutations and rearrangements modified gene sequences, but rearrangements were more numerous than local mutations in branches shorter than 290 generations, which represent 90% of the dataset. Thus, the mean rate of rearrangements over all branches turns up to be higher than the mean rate of local mutations. Beneficial mutations were also more frequent than neutral events, which is expected under a directional selection setting, but also depends on the fact that the artificial genetic code is not redundant. Neutral mutations can happen between the promoter and the start signal, but it is a rather small mutational target. Neutral mutations can also happen in intergenic regions, but those were not monitored here.

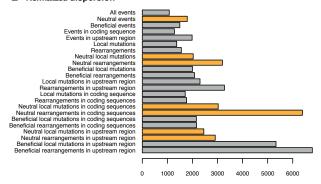
Figure 5B shows the overall normalized variation of each rate across all branches of all gene families of all runs. The indicator with the lowest normalized variation is the rate of all events that affect the gene sequence, regardless or whether they are neutral or beneficial, local mutation or rearrangement. It cannot, however, be chosen as a molecular clock, because it includes non-neutral events, whose count would be affected by the strength and type of selection. A good candidate for a molecular clock should thus both minimize its variation across branches and trees, and count neutral events only, in order to be robust to the selection regime. According to this criterion, the rate of all neutral events affecting the gene sequence, including rearrangements, would make a slightly better molecular clock than the rate of neutral local mutations only (Figure 5B, orange bars).

Analyses of gene families in real genomes of fungi, insects, and mammals have revealed a negative correlation between the age of the family and the evolutionary rate of its members (Capra et al., 2013). As shown by Figure 5C, such a negative correlation is clear in our synthetic data if all gene

A Mean rate per generation



B Normalized dispersion



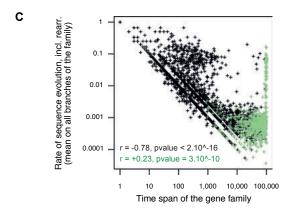


Figure 5: A. Mean evolutionary rate, per gene per generation, for each type of event (average over all branches of all gene families of all runs). B. Relative standard variation ($100 \times \text{standard deviation / mean}$) of each indicator across all branches of all gene families of all runs. Orange bars correspond to neutral indicators, candidates for a molecular clock. C. Correlation between the logarithm of family time span and the logarithm of the mean evolutionary rate in the family (black = all gene families, green = active families at t=100,000). The mean evolutionary rate of a family was computed as the average, over all its branches, of the per-generation rate of events that modify the gene without killing it. This is the indicator called "All events" in panels A and B. It includes both local mutations and chromosomal rearrangements.

families are considered (r=-0.78, p-value $< 2 \times 10^{-16}$). However, if one restricts the analysis to the observable families in the final evolved genomes, the correlation becomes positive but weaker (r=+0.23, p-value $\sim 3 \times 10^{-10}$). In the final genomes accessible to phylogenetic analyses, no trace remains from the fast evolving gene families that supported the initial (yet important) steps of the adaptation to the novel environment. Note, however, that even these final observable families evolve relatively fast and that we are actually simulating only the subset of non essential genes that confer a selective advantage in the novel environment. In contrast, in real datasets reviewed in (Capra et al., 2013), the core genes for e.g. replication and gene expression would be included. This could explain the differences in the correlation patterns between the synthetic and the real data.

Conclusion

This study of synthetic gene families revealed that, upon adaptation to a new environment, (i) there was a high turnover of gene families and extinct families showed patterns different from the final, observable ones, both in terms of dynamics of gene gains and losses and in terms of gene sequence evolution, (ii) gene sequence evolution occurred through both local mutations and chromosomal rearrangements, and (iii) incorporating chromosomal rearrangements in the evolutionary rate of gene sequences would slightly improve the accuracy of the molecular clock. Although some of the results can depend on the simplifications of the model - like the absence of redundancy of the artificial genetic code -, the study is a demonstration of how digital genetics can explore data inaccessible to classical phylogenetic methods. With refined models developed in close collaboration with phylogeneticists, digital genetics could prompt a reassessment of the biases and limitations in the studies of evolutionary dynamics of genes.

Acknowledgements

This research program was supported by the EvoEvo FP7 European project, by the PEPII program of the French CNRS and by the Rhône-Alpes Institute for Complex Systems (IXXI). We thank Eric Tannier and Guillaume Beslon for the inspiring discussions and comments.

References

Adami, C. (2006). Digital genetics: unravelling the genetic basis of evolution. *Nat. Rev. Genet.*, 7:109–118.

Akerborg, O., Sennblad, B., Arvestad, L., and Lagergren, J. (2009). Simultaneous Bayesian gene tree reconstruction and reconciliation analysis. *Proc Natl Acad Sci USA*, 106(14):5714–5719.

Arvestad, L., Berglund, A.-C., Lagergren, J., and Sennblad, B. (2004). Gene tree reconstruction and orthology analysis based on an integrated model for duplications and sequence evolution. In *Proc. RECOMB* 2004, pages 326–335. ACM Press, New York.

- Banzhaf, W. (2003). On the dynamics of an artificial regulatory network. In Banzhaf, W., Ziegler, J., Christaller, T., Dittrich, P., and Kim, J., editors, *Advances in Artificial Life*, volume 2801 of *Lecture Notes in Computer Science*, pages 217–227. Springer Berlin Heidelberg.
- Batut, B., Parsons, D., Fischer, S., Beslon, G., and Knibbe, C. (2013). In silico experimental evolution: a tool to test evolutionary scenarios. *BMC Bioinformatics*, 14(Suppl 15):S11.
- Beiko, R. G. and Charlebois, R. L. (2007). A simulation test bed for hypotheses of genome evolution. *Bioinformatics*, 23(7):825– 831.
- Beslon, G., Parsons, D. P., Sanchez-Dehesa, Y., Pena, J. M., and Knibbe, C. (2010). Scaling laws in bacterial genomes: A side-effect of selection of mutational robustness. *BioSystems*, 102(1):32–40.
- Blount, Z. D., Barrick, J. E., Davidson, C. J., and Lenski, R. E. (2012). Genomic analysis of a key innovation in an experimental Escherichia coli population. *Nature*, 489(7417):513– 518
- Boussau, B., Szollosi, G. J., Duret, L., Gouy, M., Tannier, E., and Daubin, V. (2013). Genome-scale coestimation of species and gene trees. *Genome Research*, 23(2):323–330.
- Capra, J. A., Stolzer, M., Durand, D., and Pollard, K. S. (2013). How old is my gene? *Trends in Genetics*, 29(11):659–668.
- Carvunis, A.-R., Rolland, T., Wapinski, I., Calderwood, M. A., Yildirim, M. A., Simonis, N., Charloteaux, B., Hidalgo, C. A., Barbette, J., Santhanam, B., Brar, G. A., Weissman, J. S., Regev, A., Thierry-Mieg, N., Cusick, M. E., and Vidal, M. (2012). Proto-genes and de novo gene birth. *Nature*, 487(7407):370–374.
- Dalquen, D. A., Anisimova, M., Gonnet, G. H., and Dessimoz, C. (2012). ALF–A Simulation Framework for Genome Evolution. *Molecular Biology and Evolution*, 29(4):1115–1123.
- Enright, A. J., Van Dongen, S., and Ouzounis, C. A. (2002). An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7):1575–1584.
- Frenoy, A., Taddei, F., and Misevic, D. (2013). Genetic architecture promotes the evolution and maintenance of cooperation. *PLoS Computational Biology*, 9(11):e1003339.
- Geer, L. Y., Domrachev, M., Lipman, D. J., and Bryant, S. H. (2002). CDART: protein homology by domain architecture. *Genome Research*, 12(10):1619–1623.
- Hagstrom, G. I., Hang, D. H., Ofria, C., and Torng, E. (2004). Using Avida to test the effects of natural selection on phylogenetic reconstruction methods. *Artificial life*, 10(2):157–166.
- Hang, D., Torng, E., Ofria, C., and Schmidt, T. M. (2007). The effect of natural selection on the performance of maximum parsimony. *BMC Evolutionary Biology*, 7(1):94.
- Hoban, S., Bertorelle, G., and Gaggiotti, O. E. (2012). Computer simulations: tools for population and evolutionary genetics. *Nature Reviews Genetics*, 13(2):110–122.
- Huynen, M. A. and van Nimwegen, E. (1998). The frequency distribution of gene family sizes in complete genomes. *Molecular Biology and Evolution*, 15(5):583–589.

- Jachiet, P. A., Pogorelcnik, R., Berry, A., Lopez, P., and Bapteste, E. (2013). MosaicFinder: identification of fused gene families in sequence similarity networks. *Bioinformatics*, 29(7):837–844.
- Knibbe, C., Mazet, O., Chaudier, F., Fayard, J.-M., and Beslon, G. (2007). Evolutionary coupling between the deleteriousness of gene mutations and the amount of non-coding sequences. *J. Theor. Biol.*, 244(4):621–630.
- Lin, K., Zhu, L., and Zhang, D. Y. (2006). An initial strategy for comparing proteins at the domain architecture level. *Bioin-formatics*, 22(17):2081–2086.
- Liò, P. and Goldman, N. (1998). Models of molecular evolution and phylogeny. Genome Research, 8(12):1233–1244.
- Maharjan, R. P., Gaff, J. I., Plucain, J., Schliep, M., Wang, L., Feng, L., Tenaillon, O., Ferenci, T., and Schneider, D. (2013). A case of adaptation through a mutation in a tandem duplication during experimental evolution in Escherichia coli. *BMC Genomics*, 14(1):1–1.
- Nilsson, A. I., Koskiniemi, S., Eriksson, S., Kugelberg, E., Hinton, J. C. D., and Andersson, D. I. (2005). Bacterial genome size reduction by experimental evolution. *Proc Natl Acad Sci USA*, 102(34):12112–12116.
- Parsons, D. P., Knibbe, C., and Beslon, G. (2010). Importance of the rearrangement rates on the organization of transcription. In *Proceedings of Artificial Life XII*, pages 479–486.
- Payen, C., Di Rienzi, S. C., Ong, G. T., Pogachar, J. L., Sanchez, J. C., Sunshine, A. B., Raghuraman, M. K., Brewer, B. J., and Dunham, M. J. (2014). The dynamics of diverse segmental amplifications in populations of saccharomyces cerevisiae adapting to strong selection. *G3: Genes—Genomes—Genetics*, 4(3):399–409.
- Perriere, G. and Gouy, M. (1996). Www-query: An on-line retrieval system for biological sequence banks. *Biochimie*, 78(5):364 369.
- Rasmussen, M. D. and Kellis, M. (2012). Unified modeling of gene duplication, loss, and coalescence using a locus tree. *Genome Research*, 22(4):755–765.
- Song, N., Joseph, J. M., Davis, G. B., and Durand, D. (2008). Sequence Similarity Network Reveals Common Ancestry of Multidomain Proteins. *PLoS Computational Biology*, 4(5):e1000063.
- Tenaillon, O., Rodriguez-Verdugo, A., Gaut, R. L., McDonald, P., Bennett, A. F., Long, A. D., and Gaut, B. S. (2012). The Molecular Diversity of Adaptive Convergence. *Science*, 335(6067):457–461.
- Ullrich, A., Rohrschneider, M., Scheuermann, G., Stadler, P. F., and Flamm, C. (2011). In silico evolution of early metabolism. *Artificial Life*, 17(2):87–108.
- Vilella, A. J., Severin, J., Ureta-Vidal, A., Heng, L., Durbin, R., and Birney, E. (2008). EnsemblCompara GeneTrees: Complete, duplication-aware phylogenetic trees in vertebrates. *Genome Research*, 19(2):327–335.

Jeff Clune*,1, Jean-Baptiste Mouret*,2,3 and Hod Lipson4

* These authors contributed equally to this work.

¹ University of Wyoming, Laramie, WY, USA

² Sorbonne Universités, UPMC Univ Paris 06, UMR 722, ISIR, F-75005, Paris, France

³ CNRS, UMR 7222, ISIR, F-75005, Paris, France

⁴ Cornell University, Ithaca, NY, USA.

mouret@isir.upmc.fr

Abstract of a publication in Proceedings of the Royal Society B. 2013. 280: 20122863 (Clune et al., 2013).

A long-standing, open question in biology is how populations are capable of rapidly adapting to novel environments, a trait called evolvability. A major contributor to evolvability is the fact that many biological entities are modular, especially the many biological processes and structures that can be modeled as networks, such as metabolic pathways, gene regulation, protein interactions, and animal brains. Networks are modular if they contain highly connected clusters of nodes that are sparsely connected to nodes in other clusters (Wagner et al., 2001; Leicht and Newman, 2008). Despite its importance and decades of research, there is no agreement on why modularity evolves (Wagner et al., 2001). Intuitively, modular systems seem more adaptable, a lesson well-known to human engineers, because it is easier to rewire a modular network with functional subunits than an entangled, monolithic network (Kashtan and Alon, 2005). However, because this evolvability only provides a selective advantage over the long-term, such selection is at best indirect and may not be strong enough to explain the level of modularity in the natural world (Wagner et al., 2001).

Modularity is likely caused by multiple forces acting to various degrees in different contexts (Wagner et al., 2001), and a comprehensive understanding of the evolutionary origins of modularity involves identifying those multiple forces and their relative contributions. The leading hypothesis is that modularity mainly emerges due to rapidly changing environments that have common subproblems, but different overall problems (Kashtan and Alon, 2005) (Modularly Varying Goals: MVG). It is unknown how much natural modularity MVG can explain, however, because it unclear how many biological environments change *modularly*, and whether they change at a high enough frequency for this force to play a significant role.

We investigate an alternate hypothesis that has been suggested, but heretofore untested, which is that modularity evolves not because it conveys evolvability, but as a byproduct from selection to reduce connection costs in a network (Fig. 1, Striedter (2005)). Modularity of networks is

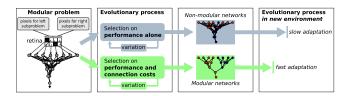


Figure 1: Main hypothesis. Evolving networks with selection for performance alone produces non-modular networks that are slow to adapt to new environments. Adding a selective pressure to minimize connection costs leads to the evolution of modular networks that quickly adapt to new environments.

measured with the Q score (Leicht and Newman, 2008).

After 25,000 generations in an unchanging environment (Fig. 2a), treatments selected to maximize performance and minimize connection costs (P&CC) produce significantly higher performing (Fig. 2c) and more modular networks (Fig. 2d) than treatments maximizing performance alone (PA) (Q = 0.42, 95%) confidence interval [0.25, 0.45] vs. $Q = 0.18[0.16, 0.19], p = 8 \times 10^{-9}$ using Matlabs Mann-Whitney-Wilcoxon rank sum test). To test whether evolved networks exhibit functional modularity corresponding to the left-right decomposition of the task, we divide networks into two modules by selecting the division that maximizes Q and color nodes in each partition differently. Left-right decomposition is visually apparent in most P&CC trials and absent in PA trials (Fig. 2e,f). Functional modularity can be quantified by identifying whether left and right inputs are in different partitions, which occurs in 56% of P&CC trials and never with PA (Fishers exact test, $p = 4 \times 10^{-11}$). Pairs of perfect sub-solution neurons – whose outputs perfectly answer the left and right subproblems – occur in 39% of P&CC trials and 0% of PA trials (Fishers exact test, $p = 3 \times 10^{-6}$).

Acknowledgements

NSF Postdoctoral Research Fellowship in Biology to JC (DBI-1003220), NSF CDI Grant ECCS 0941561, Creadapt ANR- 12-JS03-0009 to JBM.

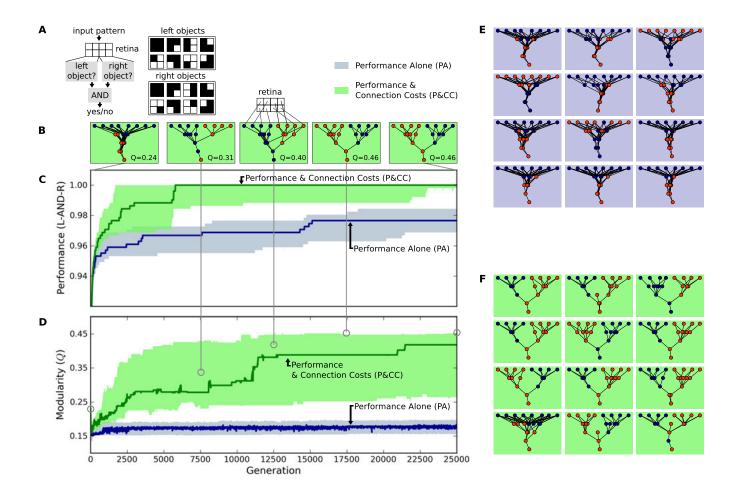


Figure 2: The addition of connection costs leads to higher-performing, functionally modular networks. (A) Networks evolve to recognize patterns (objects) in an eight-pixel retina. The problem is modularly decomposable because whether an object exists on the left and right sides can be separately determined before combining that information to answer whether objects exist on *both* sides (denoted by the AND logic function). (B) Networks from an example trial become more modular across evolutionary time with a pressure to minimize connection costs in addition to performance (P&CC). (C) Median performance (± 95% bootstrapped confidence intervals) per generation of the highest-performing network of each trial, which is perfect only when minimizing connection costs in addition to performance. (D) Network modularity, which is significantly higher in P&CC trials than when selecting for performance alone (PA). (E) The 12 highest-performing PA networks, each from a separate trial. (F) The 12 highest-performing P&CC networks, which are functionally modular in that they have separate modules for the left and right subproblems. Nodes are colored according to membership in separate partitions when making the most modular split of the network (see text).

References

Clune, J., Mouret, J.-B., and Lipson, H. (2013). The evolutionary origins of modularity. *Proceedings of the Royal Society B: Biological sciences*, 280(1755):20122863.

Kashtan, N. and Alon, U. (2005). Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences*, 102(39):13773–13778.

Leicht, E. A. and Newman, M. E. J. (2008). Community

structure in directed networks. *Physical review letters*, pages 118703–118707.

Striedter, G. (2005). *Principles of brain evolution*. Sinauer Associates Sunderland, MA.

Wagner, G., Mezey, J., and Calabretta, R. (2001). *Modularity. Understanding the development and evolution of complex natural systems*, chapter Natural selection and the origin of modules. MIT Press.

Stress-induced variation can cause average mutation and recombination rates to be positively correlated with fitness

Daniel B. Weissman^{1,2}

¹Simons Institute for the Theory of Computing and ²California Institute for Quantitative Biosciences University of California, Berkeley, CA 94720 weissman@berkeley.edu

Introduction

Observational and experimental studies have consistently demonstrated that individuals from a wide variety of species exhibit "stress-induced variation" (SIV) - higher rates of mutation and recombination in stressful conditions (see references in Hadany and Beker (2003); Hadany and Otto (2009)). This has been seen as a form of fitness-associated variation (FAV), in which an individual's mutation and recombination rates are functions of its fitness (Beker and Hadany, 2002). Theoretical studies have shown that on a variety of biologically plausible fitness landscapes, genes coding for negative associations between fitness and mutation and recombination rates ("negative FAV") are favored by evolution (Redfield, 1988; Gessler and Xu, 2000; Hadany and Beker, 2003; Hadany and Otto, 2007, 2009; Ram and Hadany, 2012); similar effects have also been found for genetic algorithms (Srinivas and Patnaik, 1994; Beker and Hadany, 2002; Rokhlenko and Wexler, 2009), although there are counterexamples (Wexler and Rokhlenko, 2007). Thus, the empirical and theoretical results seem to be consistent: low-fitness individuals tend to have higher rates of variation, and alleles which code for this effect are expected to be favored by evolution.

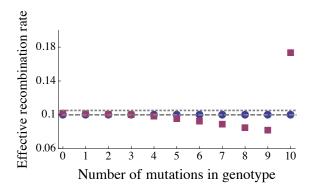
However, there is a potential issue: much of the variation in the amount of stress experienced by individuals may be caused by environmental variation rather than genetic variation, so the link between "individuals experiencing high stress" and "unfit individuals" is not completely clear. To the extent that environmental variation is completely random, with each individual experiencing small-scale fluctuations independently of every other individual, it just adds noise without disrupting the basic correlation between low genotypic fitness and high stress. But environments typically also vary in correlated ways - in particular, there are good and bad generations, where all individuals in the population feel the same decreased or increased stress, (potentially) regardless of their genotypic fitness. In fact, much of the experimental evidence for SIV comes from this kind of environmental variation, in which a population is first kept in a low-stress environment and then exposed to a stressor.

Here I show that environmental fluctuations affecting a whole population of individuals exhibiting SIV (or, more generally, population-wide changes to rates of variation) typically induce *positive* FAV – mutation and recombination rates that are positively correlated with genotypic fitness. The effect on rare genotypes with unusually high or low fitnesses is especially strong, with high-fitness genotypes having effective rates of variation many times that of low-fitness genotypes. This induced FAV can either slow or speed adaptation, depending on whether the population needs variation to occur to individuals with low fitness (to cross fitness valleys) or high fitness (to climb fitness peaks).

Model and results

Suppose that there is a population that lives in a generally mild environment but occasionally suffers stressful conditions. Under stress, an additional variation pathway is activated (e.g., a normally clonally-reproducing population undergoes sexual reproduction) or a pathway that reduces variation (e.g., kinetic proofreading) is turned off. Intuitively, during the high-variation (stressful) generations, a wide variety of genotypes are produced. These are then winnowed down by selection during the low-variation (mild) generations, so that by the time stressful conditions return, the population is composed primarily of high-fitness genotypes, which then undergo a burst of variation. This produces a bias, where higher-fitness individuals may be found more often in the high-variation generations. Note that this does not depend on the fact that increased stress was the trigger for the increase in variation; any factor that caused occasional generations with increased population-wide variation (such as increasing the mutation rate in a genetic algorithm when the rate of adaptation falls below a threshold) would have the same effect. I will stick to referring to the cause as stress, though, as it is one of the best-established mechanisms whereby this occurs.

More quantitatively, consider a period of T-1 mild generations ending with a single stressful generation. Over this time, the population as a whole experiences increased variation at an average frequency of 1/T. At what average fre-



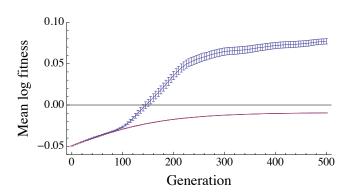


Figure 1: Comparison of populations with constant (blue) and time-varying (purple) recombination rates evolving on a 10-mutation fitness valley. (Fitness is a decreasing function of the number of mutations, except for the genotype which has all the mutations, which is the fittest). Left panel: experienced rates of recombination as a function of genotype. The dashed line shows the overall average rate, and the dotted line shows the critical rate above which the population is likely to become trapped at the lower peak (Crow and Kimura, 1965). In the variable-rate population, effective recombination rate is an increasing function of fitness. Right panel: mean log fitness of the two populations over time, with error bars showing the standard error. The population without varying recombination rates is much more successful.

quency does a particular genotype g experience it? For simplicity, assume that g is rare, has roughly constant relative fitness w_g over the T generations, and that the change in the frequency x of g is driven primarily by selection over this period. Then x(t) is simply an exponential: $x(t) \approx x(0)w_g^t$. The fraction of all g individuals alive over the T generations that experience the increased variation is $x(T)/\sum_t x(t)$, or

$$\frac{1 - w_g^{-1}}{1 - w_g^{-T}} \approx \begin{cases} 1 - w_g^{-1} & \text{for } w_g - 1 \gg 1/T \\ 1/T & \text{for } |w_g - 1| \ll 1/T \\ w_g^{T-1}(1 - w_g) & \text{for } 1 - w_g \gg 1/T. \end{cases}$$
(1)

The three different limiting cases in (1) correspond to the effect of selection on g over T generations being strongly positive, negligible, and strongly negative, respectively. Comparing them, it is easy to check that the first case has the largest effective frequency of increased variation, and the last the smallest. Note that in this limit, T does not even appear in the effective frequency of increased variation for positively-selected genotypes, suggesting that they could be disrupted even by rare stressful generations.

Effect on adaptation Perhaps the largest effect on adaptation occurs for crossing of fitness valleys (fitness landscapes in which multiple individually deleterious mutations combine to provide a fitness benefit). I show that the positive FAV induced by SIV can greatly reduce the rate of valley-crossing (Fig. 1) by increasing the effective recombination rate of the fittest genotype above the critical rate determining whether it can spread when rare. In the opposite case, a smooth landscape with many available beneficial mutations, a population with occasional bursts of recombination can adapt faster than one with a constant recombination rate (Weissman and Barton, 2012), because the induced positive FAV promotes recombination among the rare high-fitness genotypes that are driving adaptation.

References

Beker, T. and Hadany, L. (2002). Noise and elitism in evolutionary computation. *Soft Computing Systems – Design, Management and Applications*, pages 193–203.

Crow, J. F. and Kimura, M. (1965). Evolution in sexual and asexual populations. *The American Naturalist*, 99:439–450.

Gessler, D. D. and Xu, S. (2000). Meiosis and the evolution of recombination at low mutation rates. *Genetics*, 156(1):449–456.

Hadany, L. and Beker, T. (2003). Fitness-associated recombination on rugged adaptive landscapes. *Journal of Evolutionary Biology*, 16(5):862–870.

Hadany, L. and Otto, S. P. (2007). The evolution of condition-dependent sex in the face of high costs. *Genetics*, 176(3):1713–1727.

Hadany, L. and Otto, S. P. (2009). Condition-dependent sex and the rate of adaptation. *The American Naturalist*, 174(S1):S71– S78.

Ram, Y. and Hadany, L. (2012). The evolution of stressinduced hypermutation in asexual populations. *Evolution*, 66(7):2315–2328.

Redfield, R. J. (1988). Evolution of bacterial transformation: is sex with dead cells ever better than no sex at all? *Genetics*, 119(1):213–221.

Rokhlenko, O. and Wexler, Y. (2009). Embedded self-adaptation to escape from local optima. In *Proc. of the 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pages 489–496. ACM.

Srinivas, M. and Patnaik, L. (1994). Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Trans.* on Systems, Man and Cybernetics, 24(4):656–667.

Weissman, D. B. and Barton, N. H. (2012). Limits to the rate of adaptive substitution in sexual populations. *PLoS Genetics*, 8(6):e1002740.

Wexler, Y. and Rokhlenko, O. (2007). Prisoner's dilemma posed by fitness-associated recombination strategies. *Journal of Theoretical Biology*, 247(1):1–10.

Identifying Adaptations by Measuring the Trait Derivative of Fitness Extended Abstract

Drew Blount¹

¹Artificial Life Lab, Reed College, Portland, OR 97202

dblount@reed.edu

Introduction

Since Gould and Lewontin's *The Spandrels of San Marco and the Panglossian Paradigm* (1979), there has been controversy in the philosophy of biology regarding adaptationism, a research program that looks to explain evolved organisms in terms of the adaptations that comprise them. Though *Spandrels* predates the field of artificial life, its central warning should be heeded by anyone studying evolving systems, organic or artificial: it is difficult to identify adaptations.

In an evolving population where individuals have particular traits, a trait is an adaptation if it serves a function benefiting individuals' fitness, and for that reason has become prominent in the population. A much-simplified version of Gould and Lewontin's thesis is this: in general there is no way to empirically determine if a trait is an adaptation. Nonetheless, adaptationists have done so in some particular cases, generally by first conjecturing about a trait's adaptive function, and then testing other hypotheses entailed by that conjecture, confirming the adaptive premise (Dennett, 1998). It is problematic that this method can only be used when there is already a complex, plausible conjecture about a trait's adaptive function, critically limiting the set of testable traits. Further, this strategy can only conclusively confirm that a trait is an adaptation, as a negative result might only indicate that the trait has a different adaptive function than what was conjectured. There is no generalpurpose test that will tell a researcher if a trait is or is not an adaptation. This paper proposes and demonstrates such a test.

Inspired by Peter Godfrey-Smith's multi-dimensional analysis of darwinian populations (Godfrey-Smith, 2009), I analyze whether a trait is an adaptation by situating it within a three-dimensional parameter space. I measure a trait's past variability within the population, V; the trait's heritability from one generation to the next, H; and the average impact that trait has had on its host individual's reproductive fitness, the "trait-derivative of fitness," dF/dT. The necessity of considering V and H is entailed by the definition of natural selection, as I will describe. And among traits with favorable V and H values, it is those with large dF/dT values which

will, on average, become most numerous over time. Thus, there is a region of this parameter space corresponding with adaptation, and one can measure the likelihood that a trait is an adaptation by its relationship to this region.

Variability, Heritability, Differential Fitness

I will briefly describe the motivation and measurement of V, H, and dF/dT, as well as general strategies for operationalizing them. I speak only generally in this section, so that my methodology can be applied to a wide range of evolving systems.

Adaptations, by definition, only occur in populations which have previously adapted. A population can only adapt if there is variation within it, so a trait could only be an adaptation if there has previously been variation, regarding that trait, in the population of interest. To formalize this for a trait T_0 , $V(T_0)$ is a statistical measure of past variance at T_0 's locus in the evolving population.

It is clear that a trait can only be an adaptation if it is heritable – if a parent with that trait is likely to pass it on to any children. Heredity is easy to operationalize: for a trait T_0 , $H(T_0)$ is the probability that a child will have T_0 , given that its parent does.

It is often said that natural selection occurs as a combination of (at least) three things in a population: variation, heredity, and differences in reproductive fitness. This last requirement is formalized on a per-trait level in the trait-derivative: the average effect that a trait T_0 has on an individual's fitness is $\frac{dF}{dT}(T_0)$. The notation $\frac{dF}{dT}(T_0)$ is deliberately suggestive. Appro-

The notation $\frac{dF}{dT}(T_0)$ is deliberately suggestive. Appropriating the language of calculus, $\frac{dF}{dT}(T_0)$ is the expected difference in the fitness F of an organism with T_0 , if you change T_0 to $T_0 + dT$. Evaluating $\frac{dF}{dT}$ then requires operational notions of both the numerator and denominator, of fitness and trait space.

To construct or formalize a trait-space, one simply needs a measure of distance between two traits. For our purposes, this distance is closely related to H, in that $dist(T_0, T_1)$ is the probability that a parent with T_0 will have a child with T_1 . Now, the fitness of individuals within the population

must be operationalized – the exact details of this are superfluous to my argument, and different measures might be appropriate in different settings. If individual fitness can be measured or estimated well, define $F(T_0)$ as the average fitness of an individual with T_0 . $\frac{dF}{dT}(T_0)$ is then the average difference between $F(T_0)$ and $F(T_i)$, inversely weighted by $dist(T_0,T_i)$, for all traits T_i in the same trait-space as T_0 .

Identifying Adaptations in Bugs

Variation and heredity are well-understood as requirements for adaptation, and they are easily captured by standard measures of variance and probability. Because dF/dT is more novel, I will demonstrate its measurement and usefulness in an evolving population where V and H are controlled, distinguishing adaptations from non-adaped traits in Packard's Bugs artificial life model (Packard, 1989).

Bugs live in a 2d grid-world, and their 'genome' is an instruction set for where to move depending on the distribution of food within the bug's 5-cell neighborhood. Each cell either has food, or it doesn't. There are 32 possible local food maps, so there are 32 loci in a bug's genome. Genes describe movement of 1-15 steps in one of the eight cardinal directions. In my experiments, the global food map was held constant ('eaten' food does not diminish) with a centered rectangle of food covering a quarter of the world's area. In this setup, only 14 of the 32 loci in the bug genome could ever be expressed, as most 5-cell food patterns are not realized anywhere in the world map. Once each time step, every bug eats if it is standing on food, moves according to its genome and local neighborhood, and reproduces asexually if it has enough energy. Bugs lose energy when they don't eat, when they move, and when they reproduce. If a bug's energy goes below 0, it dies.

I defined the fitness of a bug at time 0 as the number of its descendants at time 1000, and the fitness of a trait as the average fitness of a bug with that trait. Genetic mutations during reproduction were locus-specific, uniformly random substitutions from trait space, so the distance between any pair of traits is equal. For this reason I did not scale by inverse distance, because that factor would be the same for each trait. $\frac{dF}{dT}(T_0)$ was then evaluated as a standardized difference between each trait's fitness and the median fitness for a trait at the same locus.

I ran 100,000 simulations with random initial populations and genomes, measuring $\frac{dF}{dT}(T_0)$ for every trait at every locus. My results illustrate the measure's usefulness, as is briefly shown in Figure 1. Bugs is a simple enough model that in many environments a human can quickly see what an 'optimal' evolutionary strategy would be - e.g., when surrounded by food, move one space (no more, because movement is expensive) in any cardinal direction. Obviously beneficial traits like these had by far the highest measured values of $\frac{dF}{dT}(T_0)$.

Genetic Locus	Outlier of	dF/dT	Move Rule
(local food map)	interest	,	(Phenotype)
	$\min \frac{dF}{dT}$	-0.9	15 NW
	$\max \frac{dF}{dT}$	328.6	1 S
	$\min \frac{dF}{dT}$	-2.8	13 NE
	$\max \frac{dF}{dT}$	8.9	14 S
	$\min \frac{dF}{dT}$	-1.3	1 W
	$\max \frac{dF}{dT}$	1.47	5 E

Figure 1: dF/dT was measured for every trait at every locus in the bug genome. Shown are the extreme dF/dT values at a very active, slightly active, and inactive locus, top to bottom. Notice the difference in magnitudes of maxima. Also note that for the active loci, the highlighted traits correspond with extremely good and bad strategies, given the environment's food distribution.

Conclusion

In the study of artificial life, as in biology, it is useful to discern which traits are adaptations. I have introduced the first general-purpose empirical test to make this classification. In this test, a trait's historical variability V, heritability H, and differential effect on fitness dF/dT are measured. Measuring these three values for a trait tells us if it is an adaptation—there is a boundary surface in this three-dimensional parameter space between adaptations and other types of traits. It will be fruitful to explore this boundary surface by plotting known adaptations against known non-adaptations, as I have shown in the ALife model Bugs. The results from Bugs are encouraging: at least in this simple system, my test is practicable and effective at identifying adaptations.

References

Dennett, D. (1998). The Leibnizian paradigm. In Hull, D. L. and Ruse, M., editors, *The Philosophy of Biology*. Oxford University Press, Oxford.

Godfrey-Smith, P. (2009). Variation, selection, and origins. In *Darwinian Populations and Natural Selection*. Oxford University Press, Oxford.

Gould, S. J. and Lewontin, R. C. (1979). The spandrels of San Marco and the Panglossian paradigm: a critique of the adaptationist programme. *Proc. R. Soc. Lond. B.*, 205:581–598.

Packard, N. H. (1989). Intrinsic adaptation in a simple model for evolution. Artificial life, 141.

Evolving Evolvability in the Context of Environmental Change: A Gene Regulatory Network (GRN) Approach

Yifei Wang¹, Stephen G. Matthews² and Joanna J. Bryson¹

¹Intelligent Systems Group, Department of Computer Science, University of Bath, Bath, BA1 7AY, U.K.

²Intelligent Systems Laboratory, Department of Engineering Mathematics, University of Bristol, BS8 1UB, U.K. yifei.wang@ieee.org

Abstract

Evolvability is the capacity of a genotype to rapidly adjust to certain types of environmental challenges or opportunities. This capacity, documented in nature, reflects foresight enabled by the capacity of evolution to capture and represent regularities not only in extant environments, but in the ways in which the environments tend to change. Here we posit that evolvability substantially benefits from the hierarchical representations afforded by Gene Regulatory Networks (GRNs). We present an extension of standard Genetic Algorithms (GAs) and demonstrate its capacity to learn a genotype phylogeny able to express rapid phenotypic shifts in the context of an oscillating environment.

Introduction

GAs are methods well-suited for search and optimisation in non-linear and high-dimensional problems. Convergence to near-optimal solutions is often perceived as the goal for GAs. However, overspecialisation can produce fragile solutions. In nature, selection also sustains *evolvability*, the capacity to change. The simplest form of evolvability is simply variation—the rate of evolution is determined by the amount of variation in a population (Fisher, 1930; Price, 1972; Reisinger and Miikkulainen, 2006; Hu and Banzhaf, 2010). More specialised evolvability can be achieved via hierarchical representations, for example single-gene control of beak length in Darwin's finches (Campàs et al., 2010). Evolvability allows rapid adaptation to regular changes in the environment, whilst also preventing premature convergence to local optima.

Here we present a system for discovering highly-evolvable genomes by exploiting GRNs (van Dijk et al., 2012; Payne et al., 2013). GRNs are already known to produce robustness via evolvability (Aldana et al., 2007a; Crombach and Hogeweg, 2008). The many-to-one mapping mechanism of genotype to phenotype implicit in GRNs enables genes to buffer against and even exploit likely variations in the genome. In addition, such a dual learning system—coupled plasticity—is known to accelerate evolution in the right contexts (Hinton and Nowlan, 1987; Kashtan et al., 2007; Borenstein and Krakauer, 2008).

Hinton and Nowlan (1987) focus on the interaction between evolution and learning, showing that coupled plasticity can solve a problem that is extremely difficult for an evolutionary process on its own. Our aim and approach here are similar, but our mechanism and outcome are novel. We explore how the robustness of GRNs can improve the evolvability of GAs by exploiting GRNs for learning structure required for quick adaptations to environmental change. This paper contributes: 1) a biologically-motivated GRN model and associated understanding of its dynamics, and 2) a demonstration of learning a quick and robust response to a changing environment—in other words, improved evolvability.

Related Work

We introduce some fundamental biological aspects of GRN, and discuss several GAs that have similar goals and properties

Gene Regulatory Networks

GRNs control the expression of genes for providing phenotypic traits in living organisms. They play a central role in cells and govern cell differentiation, metabolism, the cell cycle, and signal transduction (Karlebach and Shamir, 2008). A GRN contains a network of genes (regions of D-NA sequences) with interaction mechanisms for controlling gene expression. Epigenetic factors control the regulation of gene expression in the network without changing the genes. Shifts in gene regulation provide plasticity where organisms can adapt to environmental change. The presence of genes in the network and interactions between genes creates this plasticity.

There are a number of prior studies exploring both theoretical and practical aspects of GRNs from a biological perspective. Aldana et al. (2007b) studied the robustness and evolvability of the attractor landscape of GRNs under the process of gene duplication followed by divergence. Balleza et al. (2008) showed the criticality coupled in GRNs can generate the great diversity of dynamically robust living forms. Crombach and Hogeweg (2008) demonstrated that long-term evolution of complex GRNs in a changing environment can increase the efficiency of generating beneficial mutations.

Network Structures in Evolutionary Algorithms

Graph structures are common in Evolutionary Algorithms (EAs), such as undirected acyclic graphs (trees) in genetic programming (Koza, 1989) and cyclic graphs in evolutionary programming (Fogel, 1962). Genetic programming and grammatical evolution (Ryan et al., 1998) contain redundant genes in the chromosome, allowing some unexpressed genetic variation to be carried through generations. This redundancy can act like a memory system to enhance evolvability in dynamic environments. For example, Goldberg and Smith (1987) explored a diploid GA (with dominance operators) where a gene contains two alleles. A diploid GA was shown to adapt quicker to environmental change than a haploid (one allele per gene) GA. As well as redundancy, indirect encodings are beneficial for improving evolvability and they more closely represent the robustness and complexity found in nature (Reisinger and Miikkulainen, 2006; Reisinger et al., 2005).

Until recently, EAs had not incorporated the epigentics of GRN (Hu and Banzhaf, 2010). Lopes and Costa (2013a,b) use a model of GRN, which they refer to as an artificial regulatory network (ARN), to exploit epigenetics in genetic programming and grammatical evolution for the inverted pendulum problem, drawing artificial art, and the artificial ants problem (Sante Fe Trail problem). GRN in an EA has also been applied to financial trading (Nicolau et al., 2012). None of these GRN-based approaches to machine learning with EAs specifically address evolvability and the use of GRNs for handling environmental change. However, several authors have suggested there may be benefits to the approach (Hu and Banzhaf, 2010; Lopes and Costa, 2013a).

Model: GRN with Sexual Reproduction

The GRN with Sexual Reproduction model extends the standard GA with a GRN representation and associated reproduction and mutation operations which are different from the standard GA (see more details below). A simple model GRN provides a mechanism for improving evolvability: The interaction network is a multi-layer learning system. During the evolution process we used sexual rather than asexual reproduction in GRN because in tests it achieved better performance ¹.

GRN with Sexual Reproduction models the evolutionary process in the following steps, which are detailed in pseudocode in Algorithm 1. A large number of gene regulatory networks with a certain level of connectivity c are generated randomly; of these stable networks are selected. Then, all

the stable networks are re-selected based on their fitness. A certain number of networks are mutated according to the given mutation rate, P_m . Finally, all networks in the population undergo free combination² (Azevedo et al., 2006) during sexual reproduction.

Artificial Gene Regulatory Network

The GRN model from Wagner (1996); Siegal and Bergman (2002); Azevedo et al. (2006) is well established and provides the basic model of gene interactions required to demonstrate its capacity to learn an evolvable genotype.

For each individual in a finite population M, an $N_{qene} \times$ N_{gene} matrix W is an artificial gene network that contains the regulatory interactions among N_{gene} genes. An example is given in Figure 1A. Each element $w_{i,j}$ (i,j) $1, 2, \ldots, N_{qene}$) represents the regulatory effect on the expression of gene i of the product of gene j (see Figure 1B). The connectivity parameter c determines the proportion of non-zero elements in the network W. Through gene interactions, the regulatory effect acts on each gene expression pattern (in network W) are denoted by a state vector $\mathbf{S}(t) = (s_1(t), s_2(t), \dots, s_{N_{gene}}(t))$ where $s_i(t)$ represents the expression pattern of gene i at time t. Each value of expression state $s_i(t)$ is within the interval [-1,1] that expresses complete repression (-1) and complete activation (+1). For a given gene regulatory network W, the dynamics of S for each gene i is modelled by

$$s_i(t+1) = f\left(\sum_{j=1}^N w_{ij} s_j(t)\right),\tag{1}$$

where f(x) is a sigmoidal function. In this paper, we defined $f(x) = 2/(1+e^{-ax})-1$, where a is a free parameter determining the rate of change from complete repression to complete activation. When a is large enough, for example $a=1,000,\,f(x)$ is degenerated similarly as a sign function where f(x)=-1 for $x<0,\,f(x)=+1$ for x>0 and f(0)=0.

In all simulations, we defined the network developmental stability as the progression from an initial expression state to an equilibrium expression state (reaching a fixed pattern) by iterating Equation (1) within a fixed number of times, devT. A simple example of iterating Equation (1) is given in Figure 1C. If a given network W can achieve developmental stability, it is termed as viable or stable network, otherwise it is labelled unviable or unstable. We determined that an equilibrium expression state can be reached when the following equation is met

$$\frac{1}{\tau} \sum_{\theta=t-\tau}^{t} D\left(\mathbf{S}(\theta), \overline{\mathbf{S}}(t)\right) \le 10^{-4},\tag{2}$$

¹We did the same simulation with asexual GRN model, and found it was evolved very slow (the fitness was only improved a little during each shifting cycle) compared with sexual GRN model.

²For more details of implementation of free combination see the cited paper.

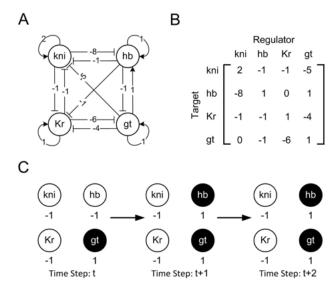


Figure 1: Example of gene regulatory network model in the gap gene system of Drosophila melanogaster, reproduced from Azevedo et al. (2006). (A), Network representation of the regulatory iteration between four gap genes (giant (gt), hunchback (hb), knirps (kni), Krüppel (Kr)) (Crawley, 2002). Gene activations and repressions are denoted by arrows and bars, respectively. Numbers on each directional edge indicate the strength (weight) of interaction between the two linked genes (Jaeger et al., 2004). (B), Interaction matrix (W) represents the network in A. Each element in row i and column j, i.e., w_{ij} , represents the regulatory effect on the expression of gene i of the product of gene j. (C), Graphical representation of the gene expression states of each gap gene over three successive time steps. We use -1 (unfilled circle) to denote the complete repression state if $s_i(t) \leq 0$, and 1 (filled circle) to represent complete activation state if $s_i(t) > 0$. Therefore, the state vectors [kni, hb, Kr, qt] of four gap genes at time t, t+1, and t+2 are S(t)=(-1,-1,-1,1), S(t+1)=(-1, 1, -1, 1), and S(t + 2) = (-1, 1, -1, 1), respectively. Successive iterations beyond the t+2 step do not change the gene expression pattern, which is the hallmark of a stable equilibrium.

where $D(\mathbf{S}, \overline{\mathbf{S}}) = \sum_{i=1}^{N_{gene}} \left(s_i - s'_i\right) / 4N_{gene}$ measures the difference between the gene expression pattern \mathbf{S} and $\overline{\mathbf{S}}$, and $\overline{\mathbf{S}}$ is the average of the gene expression levels over the time interval $[t - \tau, t - \tau + 1, \dots, t]$. Unless otherwise specified, we used devT = 100, $\tau = 10$, and a = 2 in all simulations.

Initialisation

Each individual in population M was generated with a gene regulatory network W associated with an expression state vector $\mathbf{S}(0)$. The network was generated by randomly filling

W with $c \cdot N_{gene}^2$ non-zero elements $w_{i,j} \sim N(0,1)$. The associated initial expression state $\mathbf{S}(0)$ was randomly setting each $s_i(0)$ to -1 or 1.

Selection

Competitive selection occurs using roulette-wheel selection. Survivors are placed in a new population.

Mutation

Offspring are generated by picking an individual at random from the population and placing in the new population. Then, in the selected network, each non-zero entry in the W interaction matrix was replaced by $w'_{i,j} \sim N(0,1)$ with probability P_m . Note that mutations should be viewed as operating on the $c \cdot N_{gene}^2$ cis-regulatory elements, not the coding sequences of the N_{gene} genes themselves. In other words, the mutation operation cannot change the topology of the original network W. Only offspring that are capable of producing a stable gene expression pattern survive. This process is repeated until the same amount of developmentally stable networks are produced.

Sexual Reproduction

Offspring are generated by picking two individuals at random from the population. Then the chosen two networks undergo sexual reproduction, selecting rows of the W matrices from each parent with equal probability. This process is similar to free recombination between units formed by each gene and its cis-regulatory elements, but with no recombination within regulatory regions.

Phenotype and Fitness

The fitness of an individual is determined by its phenotype. Here this is the subset of its genotype that is expressed. While in Nature the GRN might determine any number of traits to be expressed, here the fitness function considers only a fixed number (N_{PhT}) of the most upregulated genes. Level of regulation is determined by the GRN as per Equation 1. For more details of implementation of above operations please refer to Algorithm 1.

Experiments

Approach

As previously reviewed, evolvability is facilitated by redundant encoding which maintains variation, allowing quick changes to be made between good solutions. Here we have not only dual chromosomal structure, but gene activation driven by the GRN. Neither redundancy nor evolvability more generally necessarily improve EAs—in some cases these can slow evolution (Reisinger and Miikkulainen, 2006). As with all evolution and learning more generally, a gradient is required.

We compared the GRN with Sexual Reproduction model with a standard GA (Holland, 1975) to assess its capability

Algorithm 1 GRN with Sexual Reproduction Evolution Process

```
1: procedure GRN WITH SEXUAL REPRODUCTION
         set: c, devT, \tau, a
                                                                           ▶ Initialise GRN with Sexual Reproduction model parameters
 2:
                                                                                                              ▶ Initialise evolution parameters
 3:
         set: Pop_{size}, N_{gene}, N_{PhT}, P_m, G_{shift}, g and gMax
         create: \{GRN_{pop}(n); n = 1, 2, \dots, Pop_{size}\}
                                                                           \triangleright Randomly generate stable networks, using c, devT, \tau, and a
 4:
                                                   \triangleright An individual network is associated with regulatory matrix W(n) and state S(n)
         for g \leftarrow 1, gMax do
 5:
 6:
             calculate: \{Fit_{pop}(n); n = 1, 2, \dots, Pop_{size}\}
                                                                                ▶ Evaluate the fitness of each individual in the population
             while n < Pop_{size} do
 7:
 8:
                  select: GRN_{pop}(p) \ (p \in 1, 2, \dots, Pop_{size})
                                                                                                    > Select an individual based on its fitness
                  save: GRN_{pop}(n) \leftarrow GRN_{pop}(p)
                                                                                              ▶ Save the selected individual into population
 9:
10:
                  n \leftarrow n + 1
             end while
11:
             while n < Pop_{size} do
12:
13:
                  pick: GRN_{pop}(p) \ (p \in 1, 2, \dots, Pop_{size})
                                                                                                              ▶ Randomly pick one individual
14:
                  if rand(0,1) < P_m then
                                                                              \triangleright A non-zero item in W(p) is mutated with probability P_m
                      mutate: w_{i,j} \leftarrow N(0,1)
15:
                                                                                      \triangleright The non-zero item in W(p) is replaced by N(0,1)
16:
                  end if
                  if GRN_{pop}(p) is stable then
                                                                                                               ▷ Only stable networks survive
17:
                      save: GRN_{pop}(n) \leftarrow GRN_{pop}(p)
                                                                                                 ▶ Save the stable individual into population
18:
19:
                      n \leftarrow n + 1
                  end if
20:
             end while
21:
             while n < Pop_{size} do
22:
                                                                                                             ▶ Randomly pick two individuals
23:
                  pick: GRN_{pop}(p), GRN_{pop}(q)(p, q \in 1, 2, ..., Pop_{size})
                 recombine: GRN'_{pop}(p), GRN'_{pop}(q)
if GRN'_{pop}(p) and/or GRN'_{pop}(q) are stable then
save: GRN_{pop}(n) \leftarrow GRN_{pop}(p) and/or GRN_{pop}(q)
                                                                                     ▶ New individuals are generated by free combination
24:
25:
                                                                                                               > Only stable networks survive
                                                                                              ▷ Save the stable individual(s) into population
26:
27:
                      n \leftarrow n + 1 \text{ or } n \leftarrow n + 2
                                                                                                            \triangleright If both stable +2, otherwise +1
28:
                  end if
             end while
29:
             if mod(g, G_{shift}) = 0 then
                                                                                      \triangleright Test if g reach the new cycle of shifting generation
30:
                  switch fitness function
                                                                               \triangleright The environment is changed in every G_{shift} generations
31:
32:
             end if
33:
         end for
34: end procedure
```

and potential benefits for machine learning in environments that change. Here, we apply selective pressure for evolvability by changing the fitness function. We create a modified version of MaxOnes, called MaxABs. This assigns fitness according to the proportion of expressed genes that conform to the current environmental expectation, which switches between A and B after a set number of generations, which is defined by the parameter G_{shift} (see also Kashtan and Alon, 2005). For example, the first 20 generation's fitness is $\frac{number\ of\ As}{number\ of\ As+Bs}$, in the phenotype, the next 20 is $\frac{number\ of\ As}{number\ of\ As+Bs}$, the next 20 reverts to $\frac{number\ of\ As}{number\ of\ As+Bs}$, and so on. Note that for fitness, we only count As and Bs that are in the expressed genes.

Three experiments were conducted with mostly the same model parameters, but different chosen N_{qene} , N_{PhT} and

time lags (G_{shift}) between environmental changes.

Parameters

Unless specified, the mutation rate is fixed at $P_m=0.001$ for the GA and GRN with Sexual Reproduction model in all simulations. For the GA, the recombination probability P_r is fixed to be 0.7, and for the GRN with Sexual Reproduction model, free recombination is used. Network connectivity c was set to small (details below) for all runs of the GRN with Sexual Reproduction model. This is because theory shows that selection for robustness will favor more sparsely connected and minimally complex networks (Leclerc, 2008). Roulette wheel selection is used in both approaches. In all experiments, the population size was $Pop_{size}=50$. Unless specified, 500 independent runs were conducted.

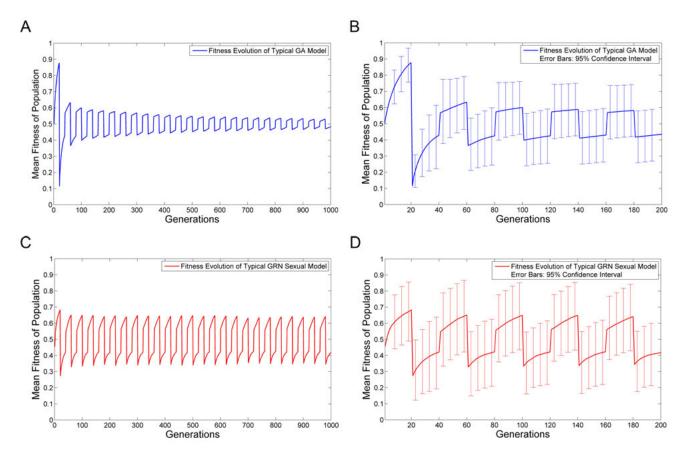


Figure 2: Evolution of both algorithms with small time gaps $(G_{shift}=20)$ between environmental change for typical GA model (A) and typical GRN with Sexual Reproduction model (C). The first 200 generations of GA and GRN with Sexual Reproduction model are shown in (B) and (D), respectively. Error bars indicate 95% confidence interval based on 500 independent runs.

Small Time Gaps Between Environmental Change

The fitness function was configured to switch between As and Bs every 20 generations to assess evolvability in frequently changing environments. The algorithms were terminated at 1,000 generations, the number of genes was $N_{gene}=20$, of which $N_{PhT}=10$ occur in the phenotype, and network connectivity was c=0.25. The mean fitness values in each population for the GA and GRN with Sexual Reproduction model are shown in Figure 2. Two observations are made.

The first observation is that the mean fitness increases to about 0.9 (for MaxAs) in the GA model, whereas it increases to about 0.7 in the GRN with Sexual Reproduction model. But for both models, the fitness does not reach the same magnitude of mean fitness in the subsequent MaxAs targets.

The second observation is that after the first two changes in fitness function (one for MaxAs and one for MaxBs) the GRN with Sexual Reproduction model has higher mean fitness than the GA. This is especially clear when the GA evolves in more cycles. The GA's mean fitness is less and

this suggests the GA is less evolvable than the GRN when switching between MaxAs and MaxBs.

Note that the goal of optimisation is to maximise the fitness value which is switching between MaxAs and MaxBs. Shifting from one peak with relatively high fitness to an even higher fitness peak is usually much more difficult than shifting from a low fitness stage to a higher fitness peak. In Figure 2, it seems that the GRN with Sexual Reproduction model has higher fitness for MaxAs and worse fitness for MaxBs. The reason is because their starting points are different. In fact, the relative increase of fitness is the same between MaxAs and MaxBs. In our simulation, we did observe that if we would give MaxBs enough time, then fitness of MaxBs can be increased to be the same level as the fitness of MaxAs.

Large Time Gaps Between Environmental Change

Based on the observation from the results produced from a small time gap, the fitness function was increased to switch between MaxAs and MaxBs every 200 generations

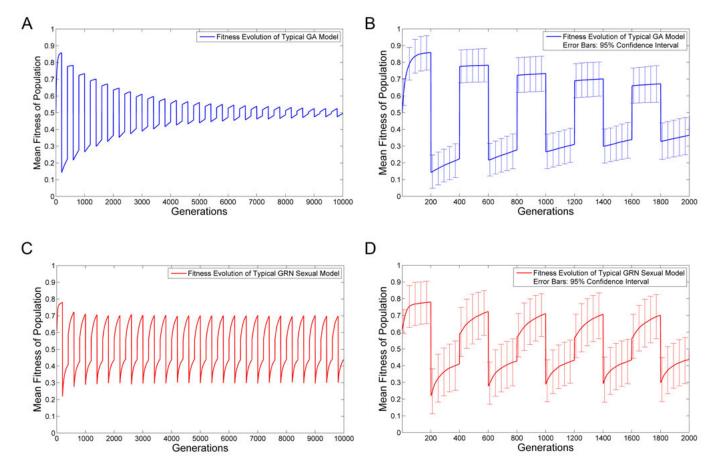


Figure 3: Evolution of both algorithms with large time gaps ($G_{shift}=200$) between environmental change for typical GA model (A) and typical GRN with Sexual Reproduction model (C). The first 2,000 generations of GA and GRN with Sexual Reproduction model are shown in (B) and (D), respectively. Error bars indicate 95% confidence interval based on 500 independent runs.

and the algorithms ran for longer by terminating at 10,000 generations. We set the phenotype to be the entire genotype, $N_{gene} = N_{PhT} = 30$, and network connectivity c = 0.05. The purpose is to assess evolvability in less frequently changing environments, but also to check the impact of redundant encoding (see experiment 3 below).

The mean fitness values in each population for the GA and GRN with Sexual Reproduction model are shown in Figure 3. Four observations are made.

The first observation is that the mean fitness of both algorithms is largest in the first time gap, which is the same for both short and long time gaps.

The second observation is that the GA has a higher mean fitness in the first time gap than the GRN with Sexual Reproduction model. This suggests the GA performs better for MaxABs *without* environmental change and the GRN with Sexual Reproduction model performs worse (perhaps learns more slowly) in a static environment.

The third observation is that the GA's mean fitness di-

minishes gradually as the fitness function is switched (Figure 3A). However, the GRN with Sexual Reproduction model achieves the same mean fitness every time the fitness function is switched (Figure 3C). This further supports that the GA is less evolvable than the GRN with Sexual Reproduction model, because the GA is less able to adapt to a regular environmental change.

The fourth observation is that the GRN with Sexual Reproduction model has steeper gradients of mean fitness in each time gap (Figure 3D). This suggests the GRN with Sexual Reproduction model is more evolvable than the GA.

Advantages of Redundant Mapping in GRN

In the first experiment, there was a redundant mapping in the GRN, but we purposely set no redundant mapping in GRN in the second experiment. Both experiments show that the GRN with Sexual Reproduction model is capable of evolving in a changing environment. However, we didn't clearly observe any advantage of redundant mapping embedded in

the GRN in short time lags. Therefore, we conducted an additional experiment to further test whether there would be a benefit of many-to-one mapping of genotype to phenotype in GRN in longer time lags.

In this experiment, the fitness function switched between MaxAs and MaxBs every 200 generations and the algorithm terminated at 5,000 generations. We set the number of genes in the genotype to $N_{gene}=20$, of which $N_{PhT}=10$ are expressed in the phenotype for selection, and network connectivity c=0.25. The result is shown in Figure 4 based on 250 independent runs. Compared to the previous experiments, two observations are made.

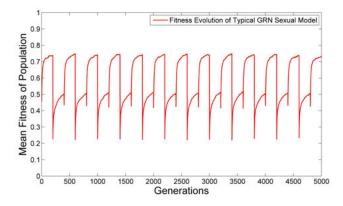


Figure 4: Evolution of typical GRN with Sexual Reproduction model with redundant mapping and large time gaps $(G_{shift}=200)$ between environmental change.

The first observation is that the GRN with Sexual Reproduction model fully recovers from the change in environment: the mean fitness in subsequent cycles can be restored to almost the same magnitude as the first cycle.

The second observation is that the GRN with Sexual Reproduction model can adapt to new environment much faster compared to Figure 2. This is particular clear when the environment switched from MaxBs to MaxAs. We further tracked how many generations would the GRN with Sexual Reproduction model be able to reach the set value of fitness³ in the first and the last cycle of MaxAs and MaxBs during the evolution. We found the GRN with Sexual Reproduction model took 150 and 103 generations on average to reach the set fitness values for MaxAs and MaxBs, respectively, in the first cycle, where as it only took 119 and 48 generations on average to reach the set fitness values in the last cycle. This is clear evidence of evolvability.

Summary & Discussion

In this paper, we show that the typical GRN with Sexual Reproduction model is capable of evolving evolvability. and thus evolving faster than the typical GA model in a changing environment. We designed two situations to mimic the changing environment, forcing the population in GA and GRN with Sexual Reproduction model to evolve in shorter and longer time lags. We found that the evolvability of the GA model decreases as it tends towards a mean value for both environments. In contrast, the GRN with Sexual Reproduction model is able to continuously evolve in both cases, retaining peak performance. In addition, although the GA model converges faster than the GRN with Sexual Reproduction model, this convergence results in a lack of diversity and evolvability. This result shows the potential advantage of using hierarchical structures such as the GRN with Sexual Reproduction model to solve problems of dynamic environments that nevertheless vary in structured ways—a task that less-structured GAs are not equipped to address.

References

- Aldana, M., Balleza, E., Kauffman, S., and Resendiz, O. (2007a). Robustness and evolvability in genetic regulatory networks. *Journal of Theoretical Biology*, 245(3):433–448.
- Aldana, M., Balleza, E., Kauffman, S., and Resendiz, O. (2007b). Robustness and evolvability in genetic regulatory networks. *Journal of Theoretical Biology*, 245(3):433–448.
- Azevedo, R. B. R., Lohaus, R., Srinivasan, S., Dang, K. K., and Burch, C. L. (2006). Sexual reproduction selects for robustness and negative epistasis in artificial gene networks. *Nature*, 440(7080):87–90.
- Balleza, E., Alvarez-Buylla, E. R., Chaos, A., Kauffman, S., Shmulevich, I., and Aldana, M. (2008). Critical dynamics in genetic regulatory networks: Examples from four kingdoms. *PLoS ONE*, 3(6):e2456.
- Borenstein, E. and Krakauer, D. C. (2008). An end to endless forms: Epistasis, phenotype distribution bias, and non-uniform evolution. *PLoS Computational Biology*, 4(10):e1000202.
- Campàs, O., Mallarino, R., Herrel, A., Abzhanov, A., and Brenner, M. P. (2010). Scaling and shear transformations capture beak shape variation in Darwin's finches. *PNAS*, 107(8):3356–3360.
- Crawley, M. (2002). Statistical Computing: An Introduction to Data Analysis Using S-Plus. Wiley.

³We set the test fitness values to be 0.6 for MaxAs, and 0.4 for MaxBs. Then we recorded the generation that the set values can be reached in 5 successive runs in a typical cycle.

- Crombach, A. and Hogeweg, P. (2008). Evolution of evolvability in gene regulatory networks. *PLoS Computational Biology*, 4(7):e1000112.
- Fisher, R. A. (1930). *The Genetical Theory of Natural Selection*. Oxford-University-Press.
- Fogel, L. J. (1962). Autonomous automata. *Industrial Research*, 4(2):14–19.
- Goldberg, D. E. and Smith, R. E. (1987). Nonstationary function optimization using genetic algorithm with dominance and diploidy. In *Proceedings of the Second International Conference on Genetic Algorithms and Their Application*, pages 59–68.
- Hinton, G. E. and Nowlan, S. J. (1987). How learning can guide evolution. *Complex Systems*, 1(3):495–502.
- Holland, J. H. (1975). Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press.
- Hu, T. and Banzhaf, W. (2010). Evolvability and speed of evolutionary algorithms in light of recent developments in biology. *Journal of Artificial Evolution and Applications*, 2010:1–28.
- Jaeger, J., Blagov, M., Kosman, D., Kozlov, K. N., Manu, Myasnikova, E., Surkova, S., Vanario-Alonso, C. E., Samsonova, M., Sharp, D. H., and Reinitz, J. (2004). Dynamical analysis of regulatory interactions in the gap gene system of drosophila melanogaster. *Genetics*, 167(4):1721–1737.
- Karlebach, G. and Shamir, R. (2008). Modelling and analysis of gene regulatory networks. *Nat Rev Mol Cell Biol*, 9(10):770–780.
- Kashtan, N. and Alon, U. (2005). Spontaneous evolution of modularity and network motifs. *PNAS*, 102(39):13773–13778.
- Kashtan, N., Noor, E., and Alon, U. (2007). Varying environments can speed up evolution. *Proceedings of the National Academy of Sciences*, 104(34):13711–13716.
- Koza, J. R. (1989). Hierarchical genetic algorithms operating on populations of computer programs. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence Volume 1*, IJCAI'89, pages 768–774, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Leclerc, R. D. (2008). Survival of the sparsest: Robust gene networks are parsimonious. *Molecular Systems Biology*, 4(1).

- Lopes, R. L. and Costa, E. (2013a). Gearnet: Grammatical evolution with artificial regulatory networks. In *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, pages 973–980.
- Lopes, R. L. and Costa, E. (2013b). Genetic programming with genetic regulatory networks: Genetic programming. In *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, pages 965–972.
- Nicolau, M., Oeill, M., and Brabazon, A. (2012). Applying genetic regulatory networks to index trading. In Coello, C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., and Pavone, M., editors, *Parallel Problem Solving from Nature - PPSN XII*, volume 7492 of *Lecture Notes in Computer Science*, pages 428–437.
- Payne, J. L., Moore, J. H., and Wagner, A. (2013). Robustness, evolvability, and the logic of genetic regulation. *Artificial Life*, 20(1):111–126.
- Price, G. R. (1972). Fisher's 'fundamental theorem' made clear. *Annals of Human Genetics*, 36(2):129–140.
- Reisinger, J. and Miikkulainen, R. (2006). Selecting for evolvable representations. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 1297–1304.
- Reisinger, J., Stanley, K. O., and Miikkulainen, R. (2005). Towards an empirical measure of evolvability. In *Proceedings of the 2005 Workshops on Genetic and Evolutionary Computation*, GECCO '05, pages 257–264
- Ryan, C., Collins, J., and O'Neill, M. (1998). Grammatical evolution: Evolving programs for an arbitrary language. In Banzhaf, W., Poli, R., Schoenauer, M., and Fogarty, T., editors, *Genetic Programming*, volume 1391 of *Lecture Notes in Computer Science*, pages 83–96. Springer Berlin Heidelberg.
- Siegal, M. L. and Bergman, A. (2002). Waddington's canalization revisited: Developmental stability and evolution. *PNAS*, 99(16):10528–10532.
- van Dijk, A. D. J., van Mourik, S., and van Ham, R. C. H. J. (2012). Mutational robustness of gene regulatory networks. *PLoS ONE*, 7(1):e30591.
- Wagner, A. (1996). Does evolutionary plasticity evolve? *Evolution*, 50(3):1008–1023.

Comparing the evolvability of generative encoding schemes

Danesh Tarapore and Jean-Baptiste Mouret

Sorbonne Universités, UPMC Univ Paris 06, UMR 722, ISIR, F-75005, Paris, France CNRS, UMR 7222, ISIR, F-75005, Paris, France daneshtarapore@gmail.com

Abstract

The evolvability of a system is the ability to generate heritable, novel and non-lethal phenotypes, from random genetic mutations. However, most evolutionary computation studies estimate evolvability either as, (i) the proportion of mutations beneficial to an individual's performance, irrespective of the phenotypic diversity of the mutants, or (ii) the range and diverseness of mutated phenotypes, without taking into account the viability of the genetic change. This paper reports a novel approach to measure the evolvability provided by an encoding, by characterizing both the quality of the mutations and the quantity of phenotypic variation. We evolved controllers for hexapod robot locomotion using a parameterized direct encoding, and the generative encoding of artificial neural networks (similar to HyperNEAT) and single-unit pattern generators (SUPGs). Our results reveal that the performance of an encoding is not always a good assessment of evolvability. Although both the generative encodings evaluated had individuals with high performance gaits, there were apparent differences in their measured evolvability. A direct and predictive relationship is indicated between our measure of evolvability, and the number of generators required by amputated individuals to recover an effective gait.

Introduction

The capacity of a population to rapidly adapt to novel environments, called evolvability, is critical in the evolutionary processes of natural organisms and artificial systems (Hu and Banzhaf, 2010). In the natural world, the high evolvability of biological systems is hypothesized to be responsible for the rich diversity of the millions of species creatively adapting to diverse niches, all arising from a combination of random mutations and natural selection (Pavlicev and Wagner, 2012). Evolvability is also of interest in evolutionary computation (EC), wherein highly evolvable solutions are considered to optimize faster and achieve a greater performance than non-evolvable solutions (e.g., Cheney et al. (2013)), and may be capable of of generalizing previously learned information and adapting it to new environments (e.g., Reisinger et al. (2005); Clune et al. (2013)), desirable characteristics from an engineering perspective.

The developmental mechanisms translating genetic change into phenotypic change, that is the genotypephenotype map, is commonly understood to play a key role in the evolvability of evolutionary systems (Kirschner and Gerhart, 1998; Wagner and Altenberg, 1996). Mappings facilitating evolvability, confer on the individual a robustness to lethal mutations, and exhibit a modular architecture wherein genes preferably only affect traits with the same function (Pavlicev and Wagner, 2012). Inspired by the evolvability of biological systems, researchers in EC have abstracted the underlying developmental processes, to formulate generative genotype-phenotype maps for artificial systems (e.g., Stanley (2007)). The resultant generative encodings frequently outperform traditional direct encodings for various application problems such as, designing 3D objects (e.g., Hornby (2005)), game playing (e.g., Reisinger and Miikkulainen (2007); Gauci and Stanley (2010)), pattern matching (e.g., Clune et al. (2011)), and robot locomotion (e.g., Hornby and Pollack (2002); Seys and Beer (2007)). Furthermore, the higher evolvability provided by generative encodings is often considered as the reason for the observed differences in performance, consequent to their capability to reuse parts of the genotype to affect different phenotypes, scale well to large phenotypic spaces, and generate modular architectures (Stanley and Miikkulainen, 2003).

In the field of EC, there are two main approaches to estimate the evolvability of artificial developmental systems. Most studies estimate evolvability either as, (i) the proportion of genetic mutations that are beneficial to an individual, irrespective of the phenotypic novelty of the resultant offspring (e.g., Hornby et al. (2003); Reisinger and Miikkulainen (2007)), or as (ii) the range and diversity of the phenotypic variants resulting from genetic change (Lehman and Stanley, 2011; Reisinger et al., 2005; Lehman and Stanley, 2013), usually without considering the deleteriousness of the change. Importantly, both these estimates when considered alone do not discount for mutations that, (i) generate very diverse phenotypes but prove lethal to an organism, and (ii) result in minor improvements to a phenotype, but are unable to generate novelty. According to the theory of facilitated variation (Gerhart and Kirschner, 2007), the capacity of an individual to evolve is a function of its ability to curtail lethal mutations, and simultaneously to decrease the number of mutations necessary to evolve diverse or novel phenotypes.

In this study we present a novel approach to visualize evolvability provided by direct and generative encodings, featuring both the quality and quantity of phenotypic variation consequent to genetic change, for hexapod robot locomotion. Evolvability is analyzed for a parametrized direct encoding, and the generative encoding of artificial neural networks (similar to HyperNEAT, Stanley et al. (2009)) and single-unit pattern generators (Morse et al., 2013), in three independent experiments. The significance of our measure of evolvability is analyzed by the ability of the robot to adapt to previously unencountered changes in its morphology.

Measuring evolvability

The evolvability provided by the direct and generative encodings is measured by computing the effect of genetic mutations on, (i) the viability of the mutated individual, and (ii) the diversity of phenotypes generated. The two resultant effects are treated separately instead of being combined into a single quantitative measure of evolvability, to consider the trade-offs between them in their individual influence on evolvability.

Feature 1: Deleteriousness of mutations. The first feature in our measure of evolvability is computed as the proportion decrement in the fitness of a mutated individual.

For an individual i and the mutant i', we have,

$$f_1 = (F_i' - F_i)/F_i (1)$$

where F_i and F'_i , are the fitness values before and after the application of a random genetic mutation, respectively.

The feature f_1 reflects the phenotype quality following beneficial ($f_1 > 0$), neutral ($f_1 \approx 0$), and deleterious ($f_1 < 0$) 0) genetic change.

Feature 2: Diversity of phenotypes. The second feature in our measure of evolvability is quantified as the difference in the phenotype, resulting from genetic mutation. There are many available diversity metrics to measure behavioral differences, ranging from the classical euclidean distance, to correlation coefficients, and various information theoretic measures (reviewed in Mouret and Doncieux (2012)). The mutual information metric, in comparison with other measures such as correlation, provides a more general criterion to investigate statistical dependences between behaviors, and is applicable to numerical and symbolic phenotypic representations.

We compute the phenotype diversity as the normalized mutual information (Cover and Thomas, 1991) between behaviors of an individual, before and after its genome is mutated. Assuming that the behavior of an individual i can be represented as a discrete vector B_i (details in Mouret and Doncieux (2012)), for the behaviors B_i and B'_i , of individual i and mutant i', we have:

$$H(B_i) = -\sum_{b_i \in B_i} p(b_i) \log p(b_i)$$
 (2a)

$$H(B_i) = -\sum_{b_i \in B_i} p(b_i) \log p(b_i)$$
(2a)

$$H(B_i, B'_i) = -\sum_{b_i \in B_i} \sum_{b'_i \in B'_i} p(b_i, b'_i) \log p(b_i, b'_i)$$
(2b)

$$f_2 = 1 - \frac{H(B_i) + H(B_i') - H(B_i, B_i')}{max(H(B_i), H(B_i')}$$
(2c)

where $H(B_i)$ is the entropy of the behavior B_i comprising the individual states b_i with probability $p(b_i)$, $H(B_i, B'_i)$ is the joint entropy between behaviors B_i and B'_i with joint probability density function $p(b_i, b'_i)$, and f_2 denotes the inverse of the normalized mutual information between the two behaviors.

The feature f_2 in our measure of evolvability represents the quantity of phenotypic variation following genetic change, and is indicative of the ability of the evolutionary system to produce novel phenotypes.

Hexapod locomotion problem

The evolution of locomotion gaits for multilegged robots is a classical problem in EC, addressed in many prior studies utilizing both direct and generative encodings (e.g., Liu and Iba (2004); Clune et al. (2011); Valsalam and Miikkulainen (2008)). In the large majority of these studies, the performance of evolved individuals is analyzed solely by the walking speed of the robot and the required number of generations of evolution. The rate of evolution and evolved performance has also been linked to evolvability provided by the encoding scheme, wherein controllers achieving a higher task fitness and requiring fewer generations to evolve are considered more evolvable (e.g., see Hornby et al. (2003); Komosiński and Rotaru-Varga (2001)). While these approaches provide interesting insights on the characteristic of the underlying genotype-to-phenotype mapping, they largely ignore its capabilities to generate viable phenotypic variations (diverse gaits in case of legged robots). However, the diversity of evolved walking gaits is important for the legged robot to recover rapidly from faults such as, the loss of one or more limbs, or motor malfunctions (Koos et al., 2013), and for the robot to adapt to previously unencountered environmental changes. Furthermore, an efficient recovery is particular relevant for hexapedal legged robots, wherein the probability of component failure is high, consequent to the large number of moving parts.

Hexapod platform details: The hexapod robot is simulated on a flat, horizontal surface (Fig. 1a), with the Open Dynamics Engine¹ (ODE) physics simulator. The robot has 18 Degrees of Freedom (DOF), 3 for each leg (Fig. 1b), and each DOF is actuated by a single servo. The first servo on each leg (s₁) actuates the horizontal orientation of the leg within range $[-\pi/8, \pi/8]$ radians. The second (s₂) and third

¹http://www.ode.org

(s₃) servos control the leg elevation and extension, respectively, each within the range of $[-\pi/4, \pi/4]$ radians.

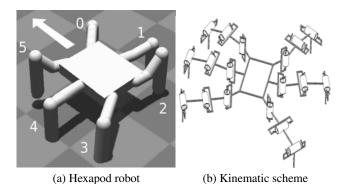


Figure 1: (a) Snapshot of an 18-DOF simulated hexapod robot walking on a horizontal surface, with contacts simulated. (b) Kinematic scheme of the robot, with cylinders representing actuated pivot joints. The three servos on each leg, s_1 , s_2 and s_3 , are labeled in increasing order of distance to robot torso.

Angular positions are sent to the 18 servos once every 15 ms. Furthermore, in order to maintain the last subsegment of each leg vertical (for enhanced stability), the control signal for the third servo (s_3) is always in antiphase to that of the second servo (s_2) . Consequently, the robot is reduced to a 12 DOF system, despite being actuated by 18 motors.

Hexapod gait representation: The phenotypic diversity in our measurement of evolvability corresponds to the intergait diversity in the hexapod robot locomotion problem. For this measurement, a hexapod gait is represented using a gait diagram (Kajita and Espiau, 2008, p. 379), comprising a binary matrix C of leg-surface contacts:

$$C_{tl} = \begin{cases} 1 & \text{if leg } i \text{ makes surface contact at time-step } t, \\ 0 & \text{otherwise.} \end{cases}$$

where $t \in \{0...T\}$, the gait is evaluated for T time-steps, and the hexapod legs $l \in \{0...5\}$.

The hexapod gait for an individual i is represented by binary vector B_i , comprising the contacts in C concatenated in row-major order, $B_i = [C_{00}, C_{10} \dots C_{T5}]$. The difference between two gaits is measured as the normalized mutual information between the corresponding gait vectors (eq. 2c).

Encoding schemes

The generative encodings for evolving hexapod locomotion controllers are based on compositional pattern producing networks (CPPNs) (Stanley, 2007). CPPNs abstract the processes of embryonic development by determining the attributes of phenotypic components as a function of their geometric location in the individual, instead of simulating complex inter-cellular interactions.

The CPPN genome is represented as a directed graph, comprising a set of *Sine*, *Gaussian*, *Sigmoid*, and *Linear* type of nodes, connected by weighted links. The node type indicates the activation function applied to the sum of its weighted inputs, to compute the node output. Selected activation functions can succinctly encode a wide variety of phenotypic patterns, such as symmetry (e.g., a Gaussian function) and repetition (e.g., a Sine function), that evolution can exploit. Mutations to the CPPN genome can change the connection weights and node type, and add or remove nodes from the graph. Consequently, the topology of the CPPN is unconstrained, and can represent any possible relationship between the input coordinates of the phenotypic component and its output attributes (see details in Stanley (2007)).

In this study, the CPPN genotype is mapped to two very different phenotypes, the conventional Artificial neural networks (ANNs), and the Single-unit pattern generators (SUPGs). The SUPG is a new type of macro-neuron introduced by Morse et al. (2013) to genetically encode spatiotemporal oscillatory patterns.

Encoding ANNs with CPPNs (minimal HyperNEAT):

The first generative encoding scheme evaluated is a simplified version of HyperNEAT indirect encoding² (Stanley et al., 2009; Clune et al., 2011; Yosinski et al., 2011). The CPPNs encode the weights of a fixed topology, single-layer feedforward ANN, featuring 2-D cartesian grids of inputs, hidden and output neurons (Fig. 2). Neurons are placed in a geometric space termed the *substrate*, so that each neuron in a layer has a distinct (x, y) coordinate. In addition, the placement of the input and output neurons is meant to reflect the hexapod robot morphology. The CPPN is iteratively input the positions of all source (x_1, y_1) and target (x_2, y_2) neurons in proximal layers, along with a constant bias, and it outputs the corresponding weights of the input-hidden and hidden-output neuron connections (see Fig. 2).

The ANN receives as input the previously requested angles (actual angles unknown) for each of the 12 pivot joints of the robot (\mathbf{s}_1 and \mathbf{s}_2 , for 6 legs). In addition, sine and cosine waves of frequency 1 Hz are also input to the ANN, to facilitate output periodic oscillations. The output from the ANN at each time-step are 12 numbers (one for each of \mathbf{s}_1 and \mathbf{s}_2 , on each of 6 legs) in interval [-1,1], that are scaled to the allowable angular range of the corresponding motors, and indicate the next position of each motor.

In preliminary experiments, this encoding evolved ANNs that exhibited high frequency output oscillations (in excess of 20 Hz). In the resultant gaits, the robot could traverse large distances by vibrating its legs rapidly, and in unison. To resolve this problem, and as suggested by Yosinski et al. (2011), we generated joint angular positions with a time-

²The CPPN is evolved with a simple multi-objective evolutionary algorithm, instead of the NEAT method (details in Tonelli and Mouret (2013)).

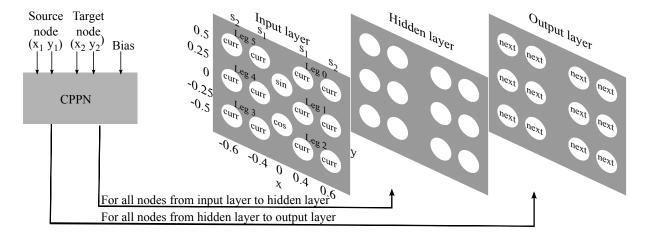


Figure 2: Encoding ANNs with CPPNs (inspired by Clune et al. (2011); Yosinski et al. (2011)). Inter-neuron connection weights are encoded as function of the position source and target neurons of each neural connection. The CPPN outputs the weights of input-hidden and hidden-output neuron connections, for each source (x_1, y_1) and target neuron (x_2, y_2) in proximal layers. The ANN is input the requested angles of the previous time-step for the first two servos $(s_1 \text{ and } s_2)$ on each leg, and a sine and cosine wave. The output neurons specify the new joint angles for the current time-step.

step of 15 ms, by averaging over four consecutive pseudopositions generated at 3.75 ms intervals. The number of evolved ANN controllers outputting high frequency oscillation was thus reduced.

Encoding SUPGs with CPPNs: In the second generative encoding scheme evaluated, the CPPN encodes the attributes of a SUPG. The SUPG is a macro-neuron (Fig. 3) that upon being triggered, produces a single cycle of a CPPN encoded activation pattern. Consequently, the repeated triggering of the SUPG results in temporal oscillations, that can be used to drive the motors of a legged robot (Morse et al., 2013).

In this encoding, the CPPN is input the position (x,y) of the SUPG in the substrate, and the elapsed time (in interval [0,1]) since the SUPG was last triggered (Fig. 3a). During the period of the SUPG, its internal timer ramps upwards with each simulation time-step, from an initial value of 0 to a maximum value of 1 (Fig. 3b). Therefore, the resultant activation pattern output by the SUPG is a function of both, its position in the substrate, and the time since the last cycle was initiated.

Applying the SUPGs for hexapod locomotion, the substrate comprises 12 SUPGs positioned to reflect the robot morphology (Fig. 3c). The outputs of the SUPGs at each time-step specify the desired angles for the first and second servos (s_1 and s_2), on each leg of the robot. The two SUPGs on each leg of the robot are triggered by the corresponding foot touching the ground, resulting in a closed-loop control. Finally, to avoid all the SUPGs being triggered simultaneously, the first trigger to each SUPG is delayed by an offset. In this study, the offset output of the CPPN is determined for the s_1 SUPG on each leg by supplying its coordinates as input, and setting the time input to 0. The same offset value

is also applied to the s_2 SUPG on the leg, allowing both the oscillators on each leg to start simultaneously.

Direct encoding: Locomotion controllers evolved with direct encoding are designed to be simple, wherein the actuation along each DOF of the robot is governed by the periodic signal of an uncoupled amplitude controlled phase oscillator. For an oscillator controlling servo \mathbf{s}_j on leg l of the robot, we have:

$$\ddot{\alpha_{lj}} = \beta \left(\frac{\beta}{4} \left(\mathbf{A}_{lj} - \alpha_{lj} \right) - \dot{\alpha_{lj}} \right)$$
 (3a)

$$\gamma_{lj} = \alpha_{lj} \cos \left(2\pi \left(t + \phi_{lj}\right)\right) \tag{3b}$$

where α_{lj} and ϕ_{lj} denote the amplitude and phase of the oscillator, A_{lj} represents its intrinsic amplitude, and γ_{lj} represents the control signal with frequency of 1 Hz. The uncoupled oscillator amplitude α_{lj} converges to the intrinsic amplitude A_{lj} at rate determined by positive constant β (set to 10 for rapid convergence). The Euler integration method with a step-size of 20 ms is used to solve the differential equation.

Hexapod leg actuation is governed by the differential equation model (eq. 3) instead of a standard periodic function (e.g., Koos et al. (2013)), to allow for smooth transitions in amplitude and frequency of the control signal between different gaits and to investigate the effects of coupling between oscillators, in our future work.

There are four evolved parameters $(A_{l1}, \phi_{l1}, A_{l2}, \phi_{l2})$ on each leg $l \in \{0...5\}$ of the robot. Consequently, a directly encoded controller for the hexapod robot is fully represented by 24 parameters.

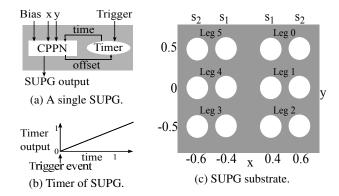


Figure 3: Encoding SUPGs with CPPNs (inspired by Morse et al. (2013)). (a) The SUPG output is a function of its coordinates (x,y) in the substrate, and the elapsed time since last trigger (output of Timer). The time of first trigger is determined by an offset. (b) Once triggered, the SUPG timer ramps upward linearly from 0 to 1 and stays there, until it is re-triggered. (c) Positions of the 12 SUPGs in the substrate, outputting the desired angles for the first two servos (\mathfrak{s}_1 and \mathfrak{s}_2), on each leg of the hexapod.

Experiments

We conducted 8000 generations of artificial selection in populations consisting of 100 individuals. Our aim was to evolve controllers for the hexapod robot to walk forward, evaluated for a period of 5 s (334 time-steps). The nondominated sorting genetic algorithm II (NSGA-II) was used to simultaneously optimize the following three objectives:

$$\text{Maximize} \begin{cases} -F_i \\ -|\Theta_i| \\ \frac{1}{N} \sum_{i=0}^{j=N} D(B_i, B_j) \end{cases}$$
 (4)

where for individual i in the population, F_i is the fitness computed as the distance between the final position of i and a goal located 25 m directly in front of the robot's initial position, Θ_i denotes the angle of the robot's trajectory with respect to the normal forward walking direction, $D(B_i, B_j)$ is the hamming distance between the binary gait vectors of individual i and j, and N is the size of the population.

In eq. 4, the first and second objectives reward individuals to walk forward large distances towards a goal, unattainable by the robot within the experiment evaluation time. The third objective is introduced to facilitate the exploration of diverse solutions and avoid premature convergence (Mouret and Doncieux, 2012).

Artificial selection was conducted in 20 independent replicates, for each of the three encodings. Performance and evolvability analysis are reported for the best individual of each replicate, selected to have the highest fitness in the population, and with an angle of trajectory in the range

of $\pm 1^{\circ}$ (simulation source code can be downloaded from http://pages.isir.upmc.fr/evorob_db.)

Individual performance

In all the encodings, the performance of the best individuals rapidly increased with a quasi-stable equilibrium being reached with less than 5000 generations of selection (Fig. 4a). Additionally, the directly encoded individuals converged more rapidly as compared to those encoded with ANN and SUPG schemes (Fig. 6a). After 8000 generations, the performance of the Direct, ANN and SUPG encodings, was 1.94 ± 0.18 , 2.93 ± 1.60 and 2.78 ± 1.43 meters, respectively (Median \pm IQR, see Fig. 4b). Amongst the three encoding schemes, the directly encoded individuals received the lowest performance (Mann-Whitney test, d.f. = 38, all p < 0.001), while no significant difference in performance was detected between the two generative encoding schemes.

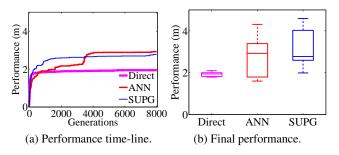


Figure 4: Performance in forward displacement for the Direct, ANN, and SUPG encodings: (a) Median performance for the 8000 generations of evolution; and (b) Performance of encodings at generation 8000.

Importantly, the frequency of the directly encoded oscillations governing leg actuation was prefixed at 1 Hz (eq. 3b). By contrast, the individuals evolved with generative encodings were capable of expressing higher frequency oscillations (2.56 ± 1.25 Hz for ANN, and 3.81 ± 0.73 Hz for SUPG), and the frequency of the gait may itself be under selection. Consequently, an assessment of the three encodings solely on the basis of the performance is biased, and other measures are needed to compare encodings.

Evolvability analysis

The evolvability provided by the encoding schemes is analyzed by mutating the best individual of each replicate at generation 8000, and reporting the following: (i) The proportion decrease in performance consequent to the mutation (eq. 1); and (ii) The gait diversity, computed as the mutual information between gait vectors of the original and mutated individual (eq. 2c). The individual is mutated 1000 times, in separate, independent instances. Finally, a kernel density estimation (Scott, 2009), is used to visualize the re-

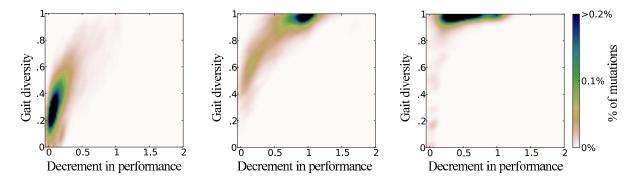


Figure 5: Gait diversity and the proportion decrease in performance, consequent to each of 20000 independent mutations, for the Direct (left), ANN (middle), and SUPG (right) encoding schemes, pooled from all 20 replicates.

sultant landscape of 20000 data points (1000 mutations \times 20 replicates), pooled together from all replicates.³

Across the three encoding schemes, the SUPG approach revealed the highest evolvability, with the capability to explore very different but viable gaits (see Fig. 5). In evolvability analysis with Direct encoding, a conservative exploration of the phenotype, limited to solutions close to the unmutated individuals was found (0.33 and 10.5\%, median gait diversity, and drop in performance, respectively). By contrast, the generative encodings were aggressive in the exploration of the phenotypic landscape, with the gait diversity of mutated individuals at 0.95 for ANN, and 0.99 for the SUPG encodings. However, differences existed in the severity of negative effects of mutations amongst the two encoding schemes. The ANN encoded individuals were sensitive to the effects of deleterious and lethal mutations⁴, resulting in a 78.9% drop in performance. In comparison, individuals evolved with the SUPG encoding were much more resilient to the negative effects of mutations, with a smaller decrement of 43.1% in performance following mutation.

Adaptation to faults

The significance of our measure of evolvability of the Direct, ANN and SUPG encodings was investigated by analyzing the adaptation of the evolved robot's gait, following the removal of one of its legs. We expect that for the encodings registering a higher measure of evolvability, the corresponding evolved individuals would require fewer generations to recoverer an effective walking gait.

In these experiments, the new populations comprised 100 mutated individuals of the best individual of each replicate at generation 8000. We conducted a further 10000 generations of artificial selection on the populations of amputee

hexapods (leg 1 removed). The number of generations required to regain an effective gait and the proportion of the original performance recovered, was analyzed.

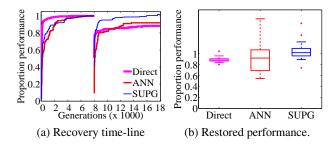


Figure 6: Recovered performance in forward displacement with the removal of leg 1 (legs numbered in Fig. 1a), for the Direct, ANN, and SUPG encodings: (a) Median proportion of performance before (left), and after (right) leg removal; and (b) Proportion of performance recovered 10000 generations after leg removal.

In the 10000 generations of selection, all the three encodings were capable of recovering the majority of their original performance (see Fig. 6a). After 10000 generations post leg removal, the recovered proportion of performance of the Direct, ANN and SUPG encodings, was 0.88 ± 0.04 , 0.92 ± 0.38 , and 1.02 ± 0.14 , respectively (Median \pm IQR, see Fig. 6b). Furthermore, amongst the three encoding schemes, the SUPG encoded individuals recovered the most of their original performance (Mann-Whitney test, d.f.=38, all p<0.001), while no significant difference in recovered performance was detected between the Direct and ANN schemes.

In order to analyze the recovery rate of ampute hexapods, in Fig. 7 we have plotted the performance lost immediately after leg removal in each of 20 replicates (horizontal axis) and the number of generations required to recover 90% of the original performance in each replicate (vertical

 $^{^3}$ Bivariate density estimation, with Gaussian type kernels over a grid of 100×100 equidistant points.

⁴Lethal mutations result in performance drops in excess of 100%, corresponding to the failure of any forward movement by the robot.

axis). The directly encoded individuals could make a 90% recovery in only 6 of 20 replicates, and suffered a median drop in performance of 60.9% immediately following leg removal. By contrast, the ANN and SUPG generative encoding schemes were better able to recover the loss in performance following amputation. In experiments with ANN encoding, 12 of 20 replicates recovered 90% of their original performance in 4392 generations (median for all replicates), despite a drastic initial loss in performance of 93.9%. The SUPG encoded amputee hexapods exhibited the fastest recovery, with 18 of 20 replicates being able to make the mark in 559 generations (median for all replicates), after incurring a smaller initial drop in performance of 67.3% consequent to leg removal.

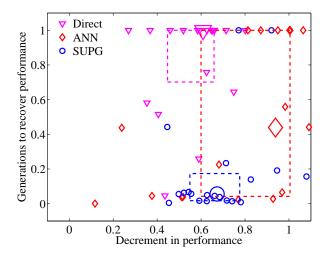


Figure 7: Proportion of performance lost immediately after leg removal and the number of generations (of 10000) required for recovery, in each of the 20 replicates (small markers), and the corresponding medians (large markers), for the Direct, ANN, and SUPG encoding schemes. The bounding boxes extend from first to third quartile.

Discussion

We investigated the evolvability provided by generatively encoded ANNs (Clune et al., 2011; Yosinski et al., 2011) and SUPGs (Morse et al., 2013), and a direct encoding, quantifying both, (i) the robustness to deleterious mutations, and (ii) the variations in the phenotypes after genetic change, for the hexapod locomotion problem. The significance of our measure of evolvability was evaluated by the individual adaptation response to morphological changes, not previously encountered by the hexapod.

Our results revealed a direct relationship between the estimated evolvability, and the capability of an individual to adapt to severe changes in its morphology, simulated by the amputation of one of its legs. Amongst the three encodings evaluated, SUPGs exhibited the highest evolvability,

and were best able to recovery following leg removal. In all but two replicates, the individuals were capable of recovering 90% of their original performance, and did so almost an order of magnitude faster than the other encodings, perhaps consequent to the closed-loop control ingrained in the SUPG encoding scheme. Such a mechanism would attempt to reform deleteriously mutated gaits, and consequently buffer the individual from the negative effects of such mutations. However, the effects of such a self-correction mechanism on evolvability needs further investigation.

The generatively encoded ANN and SUPG individuals achieved high performance gaits, when evolved for the symmetric hexapod robot. However, the performance recovery following leg removal was appreciably different between the two encoding schemes. The longer recovery of ANN individuals may be consequent to the resultant unsymmetrical and irregular hexapod morphology, a region of the problem geometry space wherein ANN generative encodings have been shown to perform poorly (Clune et al., 2011). Consequently, the damage recovery of ANN-encoded individuals would improve in situations wherein changes to the hexapod morphology preserve its symmetry (such as the removal of two middle legs). The outcome of such a scenario needs to be explored further.

For our measure of evolvability, we mutated the individuals with a predetermined mutation rate, tuned to allow a speedy convergence of the evolved solutions. This is a critical consideration, as variations to the mutation rate can affect the viability and gait diversity of generated mutants. Preliminary results suggest that with an increase in mutation rate, the peak of the distribution of mutants shifts towards more diverse gaits. However, the overall shape of the distribution, highlighting desirable regions of the evolvability landscape, remains the same for all three encodings. A rigorous analysis on the effect of variations in mutation rate and size (e.g., small mutations to many genes, or large mutations to a few genes) on the evolvability landscape, is part of our future work.

In this study, we present a novel approach to visualize the evolvability of a genotype-phenotype map, to analyze simultaneously the quality of genetic mutations, and the quantity of phenotypic variation generated. Our measure of evolvability may allow the selection of an encoding capable of producing adaptable individuals, when a inter-encoding performance comparison is no longer sufficient. In our model, the phenotypic variation was associated with the mutual information between hexapod gaits, but the diversity may be similarly computed for bipedal and quadrupedal robots, and for other qualities, such as the final position of a robot in a maze navigation task (Lehman and Stanley, 2011; Mouret and Doncieux, 2012). In this way, we hope our work may serve as a stepping stone for further research on evolvability, applied to a wide range of problems in the field of evolutionary computation.

- **Supplemental data:** Movies of hexapod walking behavior are online at http://goo.gl/4HC5MY.
- **Acknowledgment:** This study was supported by the ANR Creadapt project (ANR-12-JS03-0009).

References

- Cheney, N., MacCurdy, R., Clune, J., and Lipson, H. (2013). Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. In *Proc. of GECCO*, pages 167–174. ACM, New York, NY.
- Clune, J., Mouret, J.-B., and Lipson, H. (2013). The evolutionary origins of modularity. *Proceedings of the Royal Society b: Biological sciences*, 280(1755):20122863.
- Clune, J., Stanley, K. O., Pennock, R. T., and Ofria, C. (2011). On the performance of indirect encoding across the continuum of regularity. *IEEE Trans. Evol. Comput.*, 15(3):346–367.
- Cover, T. M. and Thomas, J. A. (1991). *Elements of information theory*. John Wiley & Sons, Inc.
- Gauci, J. and Stanley, K. O. (2010). Autonomous evolution of topographic regularities in artificial neural networks. *Neural Comput.*, 22(7):1860–1898.
- Gerhart, J. and Kirschner, M. (2007). The theory of facilitated variation. *Proc. Natl. Acad. Sci. U.S.A.*, 104(Suppl 1):8582–8589
- Hornby, G. S. (2005). Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design. In *Proc. of GECCO*, pages 1729–1736. ACM, New York, NY.
- Hornby, G. S., Lipson, H., and Pollack, J. B. (2003). Generative representations for the automated design of modular physical robots. *IEEE Trans. Robot. Automat.*, 19(4):703–719.
- Hornby, G. S. and Pollack, J. B. (2002). Creating high-level components with a generative representation for body-brain evolution. *Artif. Life*, 8(3):223–246.
- Hu, T. and Banzhaf, W. (2010). Evolvability and speed of evolutionary algorithms in light of recent developments in biology. J. Artif. Evol. App., 2010:1:1–1:28.
- Kajita, S. and Espiau, B. (2008). Springer handbook of robotics. In Siciliano, B. and Khatib, O., editors, Springer handbook of robotics, chapter 16. Springer Berlin Heidelberg.
- Kirschner, M. and Gerhart, J. (1998). Evolvability. *Proc. Natl. Acad. Sci. U.S.A.*, 95(15):8420–8427.
- Komosiński, M. and Rotaru-Varga, A. (2001). Comparison of different genotype encodings for simulated three-dimensional agents. *Artif. Life*, 7(4):395–418.
- Koos, S., Cully, A., and Mouret, J.-B. (2013). Fast damage recovery in robotics with the T-resilience algorithm. *Int. J. Robot. Res.*, 32(14):1700–1723.

- Lehman, J. and Stanley, K. O. (2011). Improving evolvability through novelty search and self-adaptation. In *Proc. of CEC*, pages 2693–2700. IEEE Press, Piscataway, NJ.
- Lehman, J. and Stanley, K. O. (2013). Evolvability is inevitable: Increasing evolvability without the pressure to adapt. *PloS one*, 8(4):e62186.
- Liu, H. and Iba, H. (2004). A hierarchical approach for adaptive humanoid robot control. In *Proc. of CEC*, volume 2, pages 1546–1553 Vol.2. IEEE Press, Piscataway, NJ.
- Morse, G., Risi, S., Snyder, C. R., and Stanley, K. O. (2013). Single-unit pattern generators for quadruped locomotion. In *Proc. of GECCO*, pages 719–726. ACM, New York, NY.
- Mouret, J.-B. and Doncieux, S. (2012). Encouraging behavioral diversity in evolutionary robotics: An empirical study. *Evol. Comput.*, 20(1):91–133.
- Pavlicev, M. and Wagner, G. P. (2012). Coming to grips with evolvability. *Evol. Educ. Outreach*, 5(2):231–244.
- Reisinger, J. and Miikkulainen, R. (2007). Acquiring evolvability through adaptive representations. In *Proc. of GECCO*, pages 1045–1052. ACM, New York, NY.
- Reisinger, J., Stanley, K. O., and Miikkulainen, R. (2005). Towards an empirical measure of evolvability. In *Proc. of GECCO*, pages 257–264. ACM, New York, NY.
- Scott, D. W. (2009). Multivariate density estimation: theory, practice, and visualization, volume 383. John Wiley & Sons, Inc.
- Seys, C. W. and Beer, R. D. (2007). Genotype reuse more important than genotype size in evolvability of embodied neural networks. In *Proc. of ECAL*, pages 915–924. Springer Berlin Heidelberg.
- Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic programming* and evolvable machines, 8(2):131–162.
- Stanley, K. O., D'Ambrosio, D. B., and Gauci, J. (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artif. Life*, 15(2):185–212.
- Stanley, K. O. and Miikkulainen, R. (2003). A taxonomy for artificial embryogeny. *Artif. Life*, 9(2):93–130.
- Tonelli, P. and Mouret, J.-B. (2013). On the relationships between generative encodings, regularity, and learning abilities when evolving plastic artificial neural networks. *PloS one*, 8(11):e79138.
- Valsalam, V. K. and Miikkulainen, R. (2008). Modular neuroevolution for multilegged locomotion. In *Proc. of GECCO*, pages 265–272. ACM, New York, NY.
- Wagner, G. P. and Altenberg, L. (1996). Perspective: Complex adaptations and the evolution of evolvability. *Evolution*, 50(3):967–976.
- Yosinski, J., Clune, J., Hidalgo, D., Nguyen, S., Zagal, J., and Lipson, H. (2011). Evolving robot gaits in hardware: the hyperneat generative encoding vs. parameter optimization. In *Proc. of ECAL*, pages 890–897. MIT Press, Cambridge.

The Evolution of General Intelligence

Padmini Rajagopalan¹, Kay E. Holekamp² and Risto Miikkulainen¹

¹University of Texas at Austin, Austin, TX 78712 ²Michigan State University, East Lansing, MI 48824 padmini@cs.utexas.edu

Abstract

When studying different species in the wild, field biologists can see enormous variation in their behaviors and learning abilities. For example, spotted hyenas and baboons share the same habitat and have similar levels of complexity in their social interactions, but differ widely in how specific vs. general their behaviors are. This paper analyzes two potential factors that lead to this difference: the density of connections in the brain, and the number of generations in prolonged evolution (i.e. after a solution has been found). Using neuroevolution with the NEAT algorithm, network structures with different connectivities were evaluated in recognizing digits and their mirror images. These experiments show that general intelligence, i.e. recognition of previously unseen examples, increases with increase in connectivity, up to a point. General intelligence also increases with the number of generations in prolonged evolution, even when performance no longer improves in the known examples. This outcome suggests that general intelligence depends on specific anatomical and environmental factors. The results from this paper can be used to gain insight into differences in animal behaviors, as well as a guideline for constructing complex general behaviors in artificial agents such as video game bots and physical robots.

Introduction

All species have special abilities that help them survive in their ecological and social niche. Such abilities can be termed *domain-specific intelligence*. For example, spotted hyenas are extremely capable hunters, whether alone or in a pack, but they are poor at tasks in new domains to which they have not been exposed. On the other hand, baboons, which share the same habitat and have similar social structure, are very good at solving new tasks that they have never seen before, such as finding food that is hidden from them by human beings. This ability to solve novel problems outside of specific cognitive domains is called *domain-general intelligence*. Thus, the extent of general intelligence is different in different species, even those in similar environments.

A prevailing theory is that domain-general intelligence emerges from the interaction of multiple processes or modules in the brain (Kaufman et al., 2011; van der Maas et al., 2006). So far, there have been few attempts to analyze or

verify this theory through computer simulations. In this paper, neuroevolution techniques were used to study the evolution of general intelligence. These simulations investigated the effects of different network structures and prolonged evolution on the ability to solve a new task never seen during evolution. Network connection densities were varied to resemble the interconnectivity of brain modules in different species. The length of prolonged evolution after the old tasks had been solved was also varied. It was discovered that the performance on the new task, i.e. general intelligence, increased with increasing connection density, up to a point. Performance also improved with prolonged evolution even though fitness in the old tasks no longer improved. These results suggest that general intelligence may emerge due to specific anatomical and environmental factors.

Neural networks were employed in these experiments because they have been used to simulate brain structures previously with success(Amit, 1992; Miikkulainen et al., 2005). In addition, neuroevolution has been used to study predatorprey interactions and the evolution of complex behaviors in animals (Rawal et al., 2010; Rajagopalan et al., 2011). Thus the insights from this paper should be useful in building better better models of intelligent behavior in the wild, as well as to develop more realistic artificial agents in video games or robotics.

This paper is organized as follows. The next section examines the biological background and prior work in analyzing the origins of general intelligence in animals. The neuroevolution architecture and algorithm used in this paper are discussed in the section after that. The two sections following that lay out the hypotheses and the experimental setup used to test them. The Results section describes the actual experiments performed as well as the results obtained.

Biological Background

Animals in their natural habitat face many different kinds of problems for which special abilities may be required. Indeed, all species exhibit specialized cognitive skills that are essential for their survival. These skills fall under the realm of domain-specific intelligence, and may be activated or in-

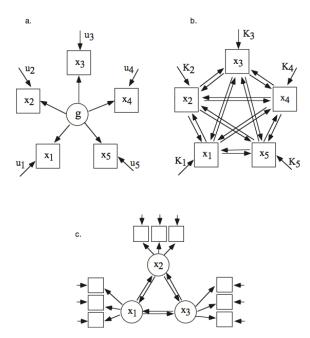


Figure 1: Three models of the positive manifold: (a) the standard g model, which shows the underlying g factor that influences all cognitive processes, (b) the mutualism model, showing the interaction between various mental mechanisms, and (c) the extended mutualism model, where only the latent processes interact. Squares and circles denote manifest and latent variables, respectively. Symbols x denote processes, y unique variances, and y resources (van der Maas et al., 2006).

hibited by specific circumstances arising in the environment. Each ability is thought to be controlled by a particular module or area of the brain (Cosmides and Tooby, 2002). Thus, many animal species excel in solving certain problems, but not others.

In contrast to domain-specific intelligence, domain-general intelligence is the ability of an individual to solve new problems. Some animal species, for example, mice (Matzel et al., 2011) and primates (Reader and Laland, 2002), possess an underlying cognitive mechanism that helps them with associative learning and general problem-solving. Evolutionary biologists and behavioral ecologists have posited that domain-general intelligence consists of several linked domain-specific abilities (Cosmides and Tooby, 2002). It has also been hypothesized that general intelligence emerged from domain-specific capabilities evolving under complex evolutionary pressures. Primatologists argue that these pressures were specifically the complex social structures and labile social interactions required of primates (Dunbar, 2003; Byrne and Whiten, 1989).

Primates are well known to be superior to most other animals in their behavioral flexibility and ability to solve prob-

lems (Byrne and Whiten, 1989; Tomasello and Call, 1997). Moreover, they live in structurally complex groups. Therefore, social cognition seems to be related to general intelligence, implying that all animal species that live in similarly large and complex societies should possess primate-like cognitive abilities. However, this prediction is not always true. For example, the societies of spotted hyenas (Crocuta crocuta) are remarkably like those of cercopithecine primates such as savanna baboons (Papio anubis) with respect to size, composition, structure, and patterns of competition and cooperation (Holekamp et al., 2012). A host of studies have found that spotted hyenas and baboons have converged with respect to social cognition (Holekamp et al., 2007; Benson-Amram et al., 2011), but there is no evidence that their domain-general intelligence has converged. This result indicates that there may be some fundamental differences in the way the brain structures and cognitive faculties of different species have evolved.

General intelligence is most apparent in human beings. There is a significant positive correlation between abilities in various cognitive tasks (van der Maas et al., 2006): A good score on one cognitive test predicts good scores on all other cognitive tests. This empirical phenomenon is called the *positive manifold*, and it is unlikely to result from a strictly modular brain (Figure 1).

There are many theories regarding the origin of the positive manifold (van der Maas et al., 2006). One hypothesis is that there must be a single underlying mechanism in the brain on which general intelligence depends (van der Maas et al., 2006; Sternberg and Grigorenko, 2002). This factor is commonly denoted as *g* (Figure 1a). Another general intelligence model is called *mutualism* (van der Maas et al., 2006). In this model, the positive manifold arises from the interaction of multiple cognitive processes in the brain (Figure 1b). The extended mutualism model (Figure 1c) posits that *g* itself arises from the interaction between several latent cognitive mechanisms in the brain. These latent variables then influence the outwardly manifest variables that constitute general intelligence (van der Maas et al., 2006).

In this paper, neuroevolution was used to evaluate the theory of mutualism in the evolution of general intelligence. Neural networks represented the brains, and the neurons in them were the cognitive mechanisms that interacted to produce general intelligence. The extent of interactions was characterized by the number of connections between neurons in a neural network. Hence, a neural network with more connections was expected to perform better when tested on a general intelligence task. In addition, the amount of task-specific learning was varied to check whether there was a positive manifold or correlation between performance in that task and performance in a new cognitive task.

NeuroEvolution of Augmenting Topologies

Neuroevolution has previously been used to great effect for producing dynamic and intelligent behaviors in autonomous agents. For example, it has been used in simulated robot soccer (Whiteson et al., 2005), robotic battle (Stanley and Miikkulainen, 2004) and Ms. Pac-Man (Burrow and Lucas, 2009). It has also been used to simulate the hunting and social behaviors of animal groups (Rawal et al., 2010; Rajagopalan et al., 2011; Rawal et al., 2012). Thus, neuroevolution is a natural choice for modeling the emergence of general intelligence.

NeuroEvolution of Augmenting Topologies, or NEAT (Stanley and Miikkulainen, 2002), is a neuroevolution technique that optimizes not only the connection weights, but also the topology of a neural network. NEAT has the resources to add and delete both nodes and links. This technique was shown to be more effective than traditional neuroevolution methods that modify only the connection weights of neural networks (Stanley and Miikkulainen, 2002).

To make crossover between two neural networks with different topologies possible, their links have to be lined up according to the nodes they connect. NEAT makes this possible using a historical marker called a global innovation number which records when each connection was formed in the population's history. Speciation is also used to nurture new innovations in network structure that might otherwise be lost due to their low initial fitnesses.

In this paper, the NEAT algorithm was used to evolve simple neural networks. These networks were trained on one cognitive task, but tested on a different task to study their ability to adapt. The number of links in the neural networks was varied to analyze the effect of connection density on performance in general intelligence, or the testing phase. It is easy to manipulate the connectivity of networks in NEAT by changing the probability of adding new links. This ability makes it a good choice as the neuroevolution algorithm for this particular study.

Hypotheses

The experiments in this paper were conducted with the objective of simulating the early evolutionary processes in the brains of species with different levels of domain-general intelligence. With this goal in mind, two factors thought to influence general intelligence were tested. The following hypotheses specify the expected results:

1. The general intelligence should increase with the connection density.

Theories from differential psychology claim that more interaction between the various modules in the brain that have evolved for specific tasks results in higher domaingeneral intelligence (van der Maas et al., 2006). In the

simulations in this paper, a neural network with more connections between its neurons is expected to have perform better in the testing phase.

The general intelligence should increase with extended evolution.

As stated in the Biological Background section, performance in various cognitive tasks are found to be positively correlated with one another. If neural networks are evolved past the point where their performance in the known tasks has plateaued, they evolve to be more diverse, robust and efficient in these tasks (Watson et al., 2011; Lehman and Stanley, 2010). They are more resilient to changes in network function (such as mutations), and should be more resilient to changes in the input as well, and thereby more general. As a result, their performance in the testing phase should improve with prolonged evolution on the first task.

Experimental Setup

This section describes the experimental setup used for the experiments in this paper. As explained in the previous sections, the goal was to computationally explore theories from neurobiology and psychology about the evolution of domain-general intelligence. To simulate the evolution of animal brains, simple two-layer neural networks were evolved using the NEAT algorithm.

The connections between the neurons simulated the interaction of processes in the brain, and the length of prolonged evolution (i.e. evolution after a satisfactory performance had been reached) represented the opportunity to evolve generalized processes. In the testing phase, new input was presented to the neural network. These inputs were related to the old inputs, but were not shown to the neural network during evolution. The performance on the new input was taken as the general intelligence of that particular neural network. Fitness was defined as the percentage of inputs for which the correct output was produced.

All neural networks in the population were identical before the start of evolution. They each had a single hidden layer of neurons and sigmoidal activation functions. In the preliminary experiments, topology as well as weights could change through evolution. The later experiments involved the evolution of only the connection weights. Even when topology was evolved, only links were added or deleted; the number of hidden neurons was always constant. The population consisted of a single species of either 30 or 100 individual neural networks. This number and the number of species changed during the program's execution as part of the NEAT algorithm. Each experiment was run 20 times for each population of neural networks and their results were averaged.

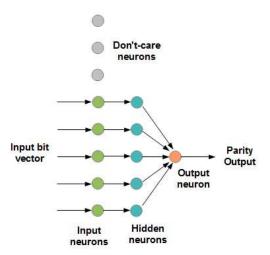


Figure 2: The initial network topology for the parity task. There are eight input nodes of which five are active, and three are don't cares. The five input nodes are connected to five hidden neurons, each of which is connected to the single output node. This preliminary task demonstrated that the topology-evolving approach would converge to similar architectures for this simple problem. Therefore, a fixed-topology approach in a more challenging task was necessary to test the hypotheses.

Results

Preliminary experiments were first conducted in the multibit parity domain to check what kinds of tasks and neural network architectures were most suitable for simulating general intelligence. In these experiments, both the connection weights and the structure of the networks were evolved using NEAT. Although links between already existing neurons were added or removed during evolution, neurons themselves were not created or deleted. Thus, the only factors affecting fitness were the number and location of the links, and the values of their weights. In this manner, interconnections between various brain modules were modeled. Insights from these experiments were then used to design a second set of experiments on a handwritten digit recognition domain.

Parity Task for Topology-Evolving Networks

For practical reasons, it is useful to start any simulation study with the simplest of tasks and the most basic neural network structures. After gaining insights on how to make them work, the algorithm and the task may be expanded gradually to more complex tasks. In this case, a simple logic function, multi-bit parity, served as a starting point to test the neural networks. In this task, the answer is 1 if the number of 1's in the input is even. The neural networks had eight neurons or nodes in the input layer, five nodes in the hidden layer and a single output neuron (Figure 2). Initially, each of the five

hidden neurons was connected to a different input neuron. Due to the sigmoidal activation function, the output neuron produced a real-valued number between 0 and 1. This value was taken to mean a 0 if less than 0.5, and a 1 otherwise. Of the eight input nodes, five were in use and had incoming multi-bit input, one bit at each node. Hence, there were 32 possible input bit strings. The other three nodes had as input fixed "don't-care" values. The neural networks were expected to ignore the don't-care inputs while successfully calculating the parity of the correct inputs.

Several combinations of evolution and testing scenarios were given to the networks and the percentage of testing phase inputs for which the network gave the wrong output (error) was recorded. Several neural network populations with different probabilities of adding links during evolution were tested in each of these. The idea was that networks with a higher probability of having links added would have acquired more links by the end of the simulation. Thus, according to our original hypotheses, these networks should perform better in the testing phase.

More specifically, the following experiments were performed in the parity task:

1. *Evolution task*: 22 of the 32 possible input bitstrings; don't care = 0

Test set: The other 10 bitstrings; don't care = 0

2. *Evolution task*: 22 of the 32 possible input bitstrings; don't care = 0

Test set: The other 10 bitstrings; don't care changes to 1

3. *Evolution task*: All 32 bit combinations; don't care = 0

Test set: All 32 bit combinations; don't care = 1

The purpose of conducting these different experiments was to evolve a neural network on one set of inputs and then test it on another, related set. If the neural network had evolved to generalize, the test problem should be solvable even if never seen before by the network. The performance on the new inputs was also expected to depend on the number of connections between the neurons of the network in question.

The main result was that all the different populations of neural networks (with the different probabilities of adding links during evolution) had similar performances during the testing phase. The reason was that all the networks had similar structure at the end of the simulation. Apparently, the NEAT algorithm is capable of adding links as needed for different network structures regardless of the probability of adding links. Moreover, although some links had been added for every network, the final structures still remained very simple. This result proved that the parity task was solvable by too simple a network structure. Therefore, a more

Figure 3: An example of the 8×8 bitmap encoding the image of a single digit, 7. The 1's represent black and the 0's represent white. This bitmap is concatenated into a 64-bit vector, which is given as input to the neural networks.

complex problem as well as a different neuroevolution strategy was needed to test the hypothesis that more connections mean more general intelligence.

Handwritten Digit Recognition Task for Fixed-Topology Networks

To avoid the problem with the parity experiments, in the second set of experiments, the topologies of all the networks were fixed and only the connection weights were evolved. Twenty trials of each experiment were run for each population and their results were averaged. The connection density remained the same between trials, but the individual connections could be different. Both feed-forward and recurrent neural networks were tested.

The recognition of handwritten digits, a well-known benchmark problem for neural networks, was chosen as a more challenging task, adapted for a test for domain-general intelligence. The inputs were not only the images of the digits, but also their mirror images. The mirror image of a single digit, 7, was left out of the set of inputs used during evolution and only used as a test input. The digit 7 was chosen for this purpose because the mirror image of 7 is not easily confused with other digits, and 7 is distinct from its mirror image (unlike 0, 1 or 8). If the neural network had evolved to recognize that some of its inputs were mirror images of other inputs, it should have no difficulty in recognizing a previously unseen set of mirror images. The performance during the testing phase is thus a good measure of the extent to which general intelligence has evolved in a neural network.

The handwritten digits were obtained from data stored in the NIST database. These images are taken from real-world

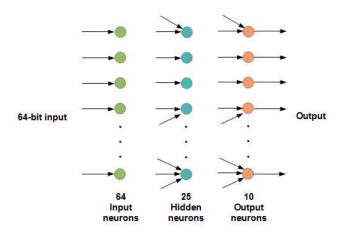


Figure 4: The network topology for the handwritten digit recognition task. There are 64 input nodes, 25 hidden neurons and ten output nodes. The connections between them are initially added at random based on the required connection density, but remain fixed thereafter. This more complex task and fixed-density architecture allowed testing the hypotheses about connectivity and prolonged evolution reliably.

data and therefore contain several noisy and distorted examples. The image of each digit was converted into an 8×8 bitmap representation, where 0 represents white and 1 represents black (Figure 3). The bits in the bitmap were then concatenated into a 64-bit vector, which was used as a single input to the neural network. Therefore, the neural networks in these experiments comprised 64 input nodes and 10 output nodes, representing a digit from 0 to 9. The digit corresponding to the node with the maximum value of output was considered the answer of the neural network to that particular input. The number of hidden neurons in each neural network was fixed at 25 (Figure 4). The number of distinct handwritten digit images shown to the networks during evolution was 5687, and the number of mirror images of 7 shown during the testing phase was 299.

The fixed topology of the neural networks meant that the effect of connection density on the testing phase performance could be easily measured. The second hypothesis, that the extent of prolonged evolution influences test performance, could also be verified by varying the number of generations for which the neural network was evolved after it had already evolved to recognize the inputs shown during evolution. The results of these experiments are shown in figures 5 and 6.

As predicted, the test performance increased with increasing connection density of the neural network. This result was true for both feed-forward and recurrent neural networks. After 65% density, however, the test performance started to decrease again (Figure 5). At the same time, the fitness on the training inputs also decreased, indicating

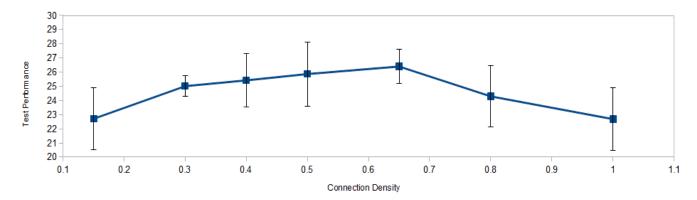


Figure 5: The average performance on the test phase (with unseen input) after evolving the network for 100 generations on the other inputs. Connection density fraction is on the *x*-axis and percentage of test input images the neural network identified correctly on the *y*-axis. The performance increases with increased connectivity up to a point, indicating more general intelligence.

that the decrease was not due to overfitting, but probably from the difficulty of evolving large numbers of connection weights.

Similarly, the test performance also increased with increasing numbers of generations for which the neural network was evolved. Evolution was continued beyond the 100 or so generations needed for networks with the optimal connection density (65%) to learn the training inputs. The performance on unseen inputs was plotted against number of generations for which the networks were evolved on the training inputs (Figure 6). It should be noted that there was no significant increase in the fitness over the evolution examples during the prolonged training. There was no effect on the results whether population size was 30 or 100. The fitness in Figure 6 for the point 0 on the *x*-axis (100 generations of evolution, stagnation just reached) is the same as the fitness for point 0.65 in Figure 5.

Thus, these results confirm that both denser connectivity and prolonged evolution result in networks that implemented more general solutions and therefore, more general intelligence.

Discussion and Future Work

The experiments described in the previous section support the hypothesis that fitness on a general intelligence task increases with increased connectivity in the brain. With the particular tasks with the simple neural networks in these experiments, more connections typically equals better generalization ability.

After a certain point, the testing performance decreases again. This is not due to overfitting but a different mechanism. The likely reason is that the neural network becomes

too complex for evolution to optimize. The networks have 64 input neurons, 25 hidden neurons and 10 output neurons each; at 0.65 connection density (i.e. with 65% of all possible connections), the number of connection weights that need to be evolved becomes very large. Excessive complexity would also explain why the populations of recurrent neural networks generally have a lower test performance than populations of feed-forward neural networks. Not only is the number of connections large, but the number of timesteps required for the network to settle is also higher.

From machine learning perspective, more connections in a neural network and prolonged training should result in overfitting, i.e. the network should not be able to generalize as well. It is interesting that the opposite effect is observed in the experiments in this paper: more connections and prolonged evolution results in more general behavior. Note that the task in this paper is not to simply interpolate between training examples, which is usually done in machine learning experiments, but to extend the behavior to a new class of examples. Such a task is more cognitive, and indeed the result is different. The task is still limited and it is difficult to demonstrate extensive general intelligence in it. But the fundamental property of general intelligence is the ability use behaviors in a novel context. Thus, it can be argued that the results indicate not just a generalization of neural networks, but the signature of general intelligence.

One significant insight gained from the above simulations is that the neural network architecture, the training algorithm, and the cognitive task employed to test general intelligence are all important. Further work is necessary to determine which combinations are most conducive to general intelligence. The main advantage of NEAT is the evo-

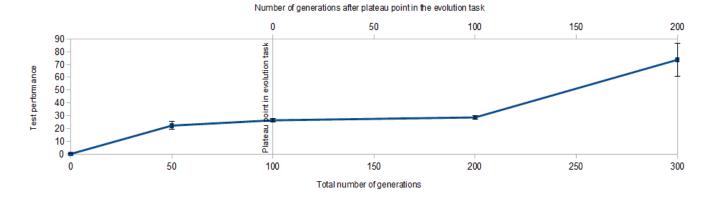


Figure 6: The average test performance of neural networks with connection density 65% after evolution for 50, 100, 200, and 300 generations on the other inputs. The training fitness usually plateaued at around 100 generations of evolution. The total number of generations is on the *x*-axis below the plot and the number of generations after fitness stagnation on the *x*-axis above the plot. The percentage of test input images the neural network identified correctly is on the *y*-axis. The performance on the test input increases with prolonged evolution.

lution of neural network structure, but since it is the level of interaction between many specialized modules that is being studied, a fixed-structure neuroevolution algorithm such as Multi-Component ESP (Gomez and Miikkulainen, 1997; Rawal et al., 2010) may be more beneficial. Different subnetworks of the same network can be evolved to perform different tasks, and their interconnections can be varied while the whole network is tested on a new domain-general task. Preliminary experiments with a hybrid NEAT-ESP algorithm are quite promising.

Similarly, the tasks themselves can be modified to resemble those faced by real animals in the wild better. Neuroevolution has already been used to evolve complex animal behaviors such as group hunting, communication with conspecifics and evading predators (Rawal et al., 2010; Rajagopalan et al., 2011; Rawal et al., 2012). Therefore, a task extension in this direction should be possible.

General intelligence approaches may ultimately be used to design robots or video game characters with realistic and complex behaviors. Such agents should be more versatile, more engaging, and more effective than is currently possible.

Conclusions

In this paper, two factors believed to influence the evolution of domain-general intelligence were tested using neuroevolution: connection density and prolonged evolution. The results indicate that increasing the number of connections leads to an increase in the general intelligence, up to a point. They also verify the hypothesis that prolonged evolution in similar cognitive tasks leads to better performance in a pre-

viously unseen task. Hence, it can be concluded that general intelligence is determined by specific anatomical and environmental factors that affect the evolution of an animal species.

The simulations from this paper can be extended in the future to more realistic tasks in environments resembling those in which real-life animal species live, gaining insight into differences observed in biology. Eventually, the same approach could be useful in creating intelligent behaviors for artificial agents in video games or robotics as well.

Acknowledgements

This research was supported in part by NSF grants DBI-0939454 and IIS-0915038, and in part by NIH grant R01-GM105042.

References

Amit, D. J. (1992). Modeling brain function: The world of attractor neural networks. Cambridge University Press.

Benson-Amram, S., Heinen, V. K., Dryer, S. L., and Holekamp, K. E. (2011). Numerical assessment and individual call discrimination by wild spotted hyaenas, (*Crocuta crocuta*). *Anim. Behav.*, 82(4):743–752.

Burrow, P. and Lucas, S. M. (2009). Evolution versus temporal difference learning for learning to play Ms. Pac-man. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG 2009)*, pages 53–60.

Byrne, R. and Whiten, A. (1989). *Machiavellian intelligence: social expertise and the evolution of intellect in monkeys, apes, and humans*. Oxford Science Publications.

- Cosmides, L. and Tooby, J. (2002). Unraveling the enigma of human intelligence: Evolutionary psychology and the multimodular mind. *The evolution of intelligence*, pages 145–198.
- Dunbar, R. I. M. (2003). The social brain: mind, language, and society in evolutionary perspective. Ann. Rev. Anthropol., 32:163–181.
- Gomez, F. and Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior*, 5(3-4):317–342.
- Holekamp, K. E., Sakai, S. T., and Lundrigan, B. L. (2007). Social intelligence in the spotted hyena (*Crocuta crocuta*). Phil. Trans. Roy. Soc. B., 362(1480):523–538.
- Holekamp, K. E., Smith, J. E., Strelioff, C. C., Horn, R. C. V., and Watts, H. E. (2012). Society, demography and genetic structure in the spotted hyena. *Molec. Ecol.*, 21(3):613–632.
- Kaufman, S. B., DeYoung, C. G., Reis, D. L., and Gray, J. R. (2011). General intelligence predicts reasoning ability even for evolutionarily familiar content. *Intelligence*, 39(5):311–322.
- Lehman, J. and Stanley, K. O. (2010). Revising the evolutionary computation abstraction: minimal criteria novelty search. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 103–110. ACM.
- Matzel, L. D., Wass, C., and Kolata, S. (2011). Individual differences in animal intelligence: learning, reasoning, selective attention and inter-species conservation of a cognitive trait. *Int. J. Comp. Psychol.*, 24:36–59.
- Miikkulainen, R., Bednar, J. A., Choe, Y., and Sirosh, J. (2005).

 Computational Maps in the Visual Cortex. Springer, New York
- Rajagopalan, P., Rawal, A., Miikkulainen, R., Wiseman, M. A., and Holekamp, K. E. (2011). The role of reward structure, coordination mechanism and net return in the evolution of cooperation. In *Proceedings of the IEEE Conference on Compu*tational Intelligence and Games (CIG 2011), pages 258–265.
- Rawal, A., Rajagopalan, P., and Miikkulainen, R. (2010). Constructing competitive and cooperative agent behavior using coevolution. In Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG 2010), pages 107–114
- Rawal, A., Rajagopalan, P., Miikkulainen, R., and Holekamp, K. (2012). Evolution of a communication code in cooperative tasks. In *Artificial Life*, volume 13, pages 243–250.
- Reader, S. M. and Laland, K. N. (2002). Social intelligence, innovation, and enhanced brain size in primates. *PNAS*, 99(7):4436–4441.
- Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.
- Stanley, K. O. and Miikkulainen, R. (2004). Competitive coevolution through evolutionary complexification. *J. Artif. Intell. Res.*, 21:63–100.
- Sternberg, R. J. and Grigorenko, E. L. (2002). *The general factor of intelligence: How general is it?* Psychology Press.

- Tomasello, M. and Call, J. (1997). *Primate cognition*. Oxford University Press, Oxford, UK.
- van der Maas, H. L. J., Dolan, C. V., Grasman, R. P. P. P., Wicherts, J. M., Huizenga, H. M., and Raijmakers, M. E. J. (2006). A dynamical model of general intelligence: The positive manifold of intelligence by mutualism. *Psychological Review*, 113(4):842–861.
- Watson, R. A., Palmius, N., Mills, R., Powers, S. T., and Penn, A.
 (2011). Can selfish symbioses effect higher-level selection?
 In Advances in Artificial Life. Darwin Meets von Neumann, pages 27–36. Springer.
- Whiteson, S., Kohl, N., Miikkulainen, R., and Stone, P. (2005). Evolving soccer keepaway players through task decomposition. *Machine Learning*, 59(1-2):5–30.

Reflective Grammatical Evolution

Christopher Timperley Susan Stepney

York Centre for Complex Systems Analysis, University of York, UK, YO10 5DD ct584@york.ac.uk susan.stepney@york.ac.uk www.yccsa.org

Abstract

Our long term goal is to develop an open-ended reflective software architecture to support open-ended evolution. Here we describe a preliminary experiment using reflection to make simple programs evolved via Grammatical Evolution robust to mutations that result in coding errors.

We use reflection in the domain of grammatical evolution (GE) to achieve a novel means of robustness by autonomously repairing damaged programs, improving continuity in the search and allowing programs to be evolved effectively using *soft* grammars. In most implementations of GE, individuals whose programs encounter errors are assigned the worst possible fitness; using the techniques described here, these individuals may be allowed to continue evolving.

We describe two different approaches to achieving robustness through reflection, and evaluate their effectiveness through a series of experiments carried out on benchmark regression problems. Results demonstrate a statistically significant improvement on the fitness of the best individual found during evolution

Keywords: reflection, open-ended evolution, grammatical evolution, robustness

Introduction

Our long term goal is to develop an open-ended reflective software architecture to support open-ended evolution *in silico*, as outlined in Stepney and Hoverd (2011). As a first step, we are making evolutionary systems more robust to evolved programs that contain errors. Here we describe an experiment using reflection to make simple programs evolved via Grammatical Evolution robust to mutations that result in coding errors.

Grammatical evolution (GE) is a metaheuristic search method belonging to the family of evolutionary algorithms. The evolutionary algorithm finds solutions to problems by evolving a population of individuals, each representing a potential solution to the given problem. In GE the individual genomes are typically represented as integer lists, decoded through a provided grammar into syntactically correct computer program phenotypes. Evolution operators may nevertheless produce semantically faulty programs. If these

faulty programs are removed from the population, any implicit heuristic information they may have gained will be lost to future generations. The alternative, examining the program phenotype to fix errors, is both complicated and grammar-dependent.

We instead use *reflection*, the ability of a program to inspect and modify its own code at run-time, to autonomously correct or *heal* the errors or *defects* that occur during the evolutionary process, in a grammar-independent manner. By being able to tolerate defects, a layer of *robustness* is incorporated into the search, allowing a wider range of mutations to be performed with less risk of losing good individuals from the population. This also allows less restrictive *softer* grammars to be used, which may improve locality in the search; traditionally such grammars would produce too many errors to be used effectively.

We demonstrate that reflection offers a viable mechanism for automatically correcting the dynamic population of individuals in GE. Using reflection, rather than a complicated grammar-dependent analysis of an individual's genotype and phenotype, allows the evolutionary process to remain unaware of the errors within its individuals' phenotypes and independent of how those errors are corrected. We achieve this isolation through a decentralised *robustness layer* that logically rests on top of the virtual machine of the programming language and below the layer of its programs.

The structure of the rest of this paper is as follows. We provide relevant contextual material on GE and reflection. We next describe two robustness layer designs, a global and a local one, to provide a robustness layer in Ruby programs. We then describe some evolutionary experiments, comparing these apporaches with a control case of no robustness layer, where faulty phenotypes are discarded.

Background

Grammatical Evolution

Grammatical Evolution (Ryan et al., 1998) is an evolutionary computation technique, implemented as a variant of the Genetic Algorithm (GA). The genotype typically takes the form of a list of integers, and the phenotype is produced by mapping the genotype to a derivation of a given grammar. Typically this grammar is used to model the grammar of a given programming language and the resulting derivations form syntactically valid programs in that language.

Unlike Genetic Programming (GP), another EA variant that directly represents and operates on individuals as computer programs (typically using a tree-like structure), GE's genotype separates the program structure from the evolutionary process. This allows programs to be evolved using arbitrary forms and languages, beyond LISP-style tree structures, and allows the GE representation to be used with non-evolutionary meta-heuristics to produce programs (Dempsey et al., 2009), such as particle swarm optimisation (O'Neill and Brabazon, 2006).

Mapping Process Individuals are mapped to their phenotype through the use of a Backus-Naur Form grammar. The values v from the list of integers are sequentially read off from left-to-right and used to select the next production rule to use from the grammar; at each step, the rule at index i is selected, where $i = v \mod m$, and m is the number of production rules available.

This process of derivation continues until one of the following situations occur:

- The mapping is completed, and all non-terminals have been transformed into terminals, producing a complete phenotype.
- The list of integers has been exhausted, but the mapping is incomplete. Usually in this case, the list is wrapped and its alleles are reused, allowing the derivation process to continue (Dempsey et al., 2009); however, this reduces locality within the chromosome, since codons may possess different meanings. An alternative is to cease the mapping process, but this produces a large number of incomplete and invalid derivations. Another alternative is to sample new codon values when they are required and to append these values to the end of the genome.
- A critical number of codons have been consumed. The derivation process is halted and the individual is assigned the worst possible fitness value; this mechanism prevents extremely long derivations from occurring.

Grammars Traditionally, GE uses heavily restricted, or *hard*, grammars designed to ensure syntactically and, as far as possible, semantically valid programs are always produced. Often these grammars are tailored to the domain of the specific problem being solved.

The performance of GE is highly sensitive to the choice of grammar down to its finest details (O'Neill et al., 2001). This problem, combined with the large domain of possible grammars, leads practitioners to use ad-hoc measures to design their grammar, if such considerations are given at all. Whilst *hard* grammars restrict the domain of the search, and

hence reduce the scale of the problem, their inflexibility can leave them trapped in a local optima and unable to escape.

We propose instead the use of *soft* grammars, that allow smaller changes to be made to the overall program. These grammars result in a larger number of semantically incorrect programs, however, so we combine this approach with a *self-repair* mechanism. This broadens the genetic land-scape and introduce flexibility into the phenotype. By incorporating such flexibility into the phenotype, neutral genetic drift is made possible; small changes to genotype may occur without dramatic consequence to the fitness of the individual. This transformation of the genetic landscape may allow the search to avoid the local optima of *hard* grammars, and may ultimately improve the performance of the algorithm.

Autonomic Computing

Self-healing is a form of *autonomic computing*, a term introduced by IBM in 2001 (Kephart and Chess, 2003), and used to describe computer systems capable of inspecting and modifying their behaviour in response to changing requirements. Inspiration for these systems came from the autonomic nervous system of the human body; an incredibly complex system composed of many interconnected components that seamlessly manage themselves to hide their complexity from us.

Kephart and Chess (2003) give four functional features that an autonomic computing system should possess:

- Self-optimisation: The ability to profile the usage and control the allocation of resources, to provide optimal service quality. Self-optimisation could also include rewriting software functions to reduce resource usage.
- Self-configuration: The ability to automatically configure components according to the context and environment of the system.
- Self-healing: The ability to automatically diagnose and correct faults.
- 4. **Self-protection:** The ability to recognise and protect itself from malicious attacks.

Whilst the grand vision of autonomic computing is yet to come to fruition, its individual features are beginning to be realised. Much of this effort has been directed towards the self-healing aspect of autonomic computing, in the field of *resilient software*, which seeks to address the inherent fragility suffered by all computer systems, by developing simple and lightweight alternatives to the difficult robustness and fault tolerance techniques employed in safety-critical system, which often involve additional hardware.

Adaptive Software Systems Haydarlou et al. (2005) proposes the use of *adaptive software systems*, which modify their own behaviour in response to errors and environmental

changes with the help of an adaptive plan describing what modifications should be made. Where such systems have been used, they have typically employed a fixed adaptive plan, pre-designed by a human programmer. An alternative, outlined in Haydarlou et al. (2005), involves using genetic programming to evolve an adaptive plan to allow the system to autonomously adapt itself to new situations.

Fraglets Tschudin (2003) describes a radically different approach to creating a robust environment for hosting resilient software, capable of addressing the problems of soft faults and program adaptability from a pure software perspective, without the need for hardware fault tolerance measures. This is achieved by using a novel software execution model inspired by the interaction of chemical molecules, which places the ability to self-modify at the core of its design.

This model is composed of small computational atoms called *Fraglets*, which when combined through reactions, process both code and data. Each time program code is required, a new instance is dynamically loaded and executed rather than being stored in the memory of the computer. This allows the code to be easily manipulated by the program without the problems of accessing live code in the memory. Lost and failed Fraglet executions are handled by creating and executing a new instance of the affected Fraglet.

Metaprogramming and Reflection

Metaprogramming is the technique of writing computer programs which manipulate or write the code of other programs (Perrotta, 2010). These meta-programs are written using a *metalanguage*, which facilities the writing and manipulation of programs written in the target *object language* (Stump, 2009).

Reflection is the ability of an object language to act as its own metalanguage (Smith, 1982; Maes, 1987; Stump, 2009; Ferber, 1989). Reflective systems may exploit the ability of reflection to inspect and manipulate the functionality and implementation of the system itself. In some programming languages, such as Lisp and Ruby, this reflective system may be the programming language itself, allowing many aspects of the implementation and semantics of the language to be inspected, extended and redefined at run-time.

Design

Here we describe two designs for implementing a robustness layer through reflection: a centralised or "global" approach, and a "local" approach. We have implement both, and compared them through evolutionary experiments.

Following an analysis of several popular reflective programming languages, we have concluded that Ruby offers the richest suite of reflective capabilities and a natural tilt towards meta-programming, making it an ideal language to explore the robustness layer approach.

ARITHMETIC FUNCTIONS

```
⟨op⟩ ::= 'add' | 'sub' | 'mul' | 'div'
# SOFT GRAMMAR
⟨program⟩ ::= ⟨block⟩
⟨block⟩ ::= ⟨fcall⟩ | ⟨float⟩ | ⟨var⟩
⟨fcall⟩ ::= ⟨string⟩ '(' ⟨fargs⟩ ')'
⟨fargs⟩ ::= ⟨block⟩ ', '⟨fargs⟩ | ⟨block⟩
⟨string⟩ ::= ⟨char⟩⟨string⟩ | ⟨char⟩ | ⟨char⟩
⟨float⟩ ::= ⟨nz_digit⟩⟨integer⟩ '.' ⟨integer⟩
| ⟨digit⟩ '.' ⟨integer⟩
⟨integer⟩ ::= ⟨digit⟩⟨integer⟩ | ⟨digit⟩
⟨char⟩ ::= 'a'..'z'
⟨digit⟩ ::= 'a'..'9'
⟨nz_digit⟩ ::= '1'..'9'
⟨var⟩ ::= 'x' | 'y'
```

Figure 1: Soft grammar used throughout paper.

```
 \begin{array}{l} sdd(afg(mua(x, mua(x, x, y)), juli(x, x)), \, x) \rightarrow \\ add(add(mul(x, mul(x, x)), mul(x, x)), \, x) \rightarrow \\ x^3 + x^2 + x \end{array}
```

Figure 2: An example of an evolved program, its effective form when using robustness measures, and its equivalent symbolic expression.

Here we investigate the use of these robustness measures with Ruby programs created via GE using a soft grammar (Figure 1). The only methods actually supported are the arithmetic functions (Figure 1), which take two arguments. The soft grammar can produce programs with method names that are arbitrary character strings, taking different numbers of arguments. (The long-term aim is to allow methods to mutate gradually into different methods; this is a first study.) To support this grammar, the robustness measures handle missing method errors, by mapping to an existing method, and incorrect argument errors, by truncating or padding the argument list. An example of a program evolved using the soft grammar is given in Figure 2.

Global Approach

The global approach achieves robustness by autonomously intercepting all exceptions thrown by monitored methods and applying fixes aimed at removing the root cause of the problem, calculated using details of the fault. This process can be seen as a form of *self-healing*, where the robustness layer *diagnoses* problems based on their details and automatically *heals* those problems by modifying the *damaged*

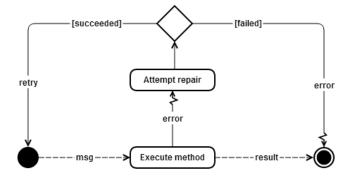


Figure 3: An overview of the global robustness layer design.

source code.

All calls to a *protected* method are routed through the global robustness layer, which attempts to execute the method according to its normal semantics.

- In the event of a fatal exception, the robustness layer becomes active and diagnoses the fault before calculating a suitable candidate fix to the issue.
- After the fix has been applied, the kernel attempts to call
 the modified method once again. In the event a subsequent error occurs, the process of diagnosis and repair is
 repeated, until either the method no longer produces errors during its execution, or till a maximum number of
 successive repairs has been reached.

Modifications made to the affected method during its "protected execution" may then be saved by the original method, removing the need to repair the method for successive calls. This ability proves useful for problems where the candidate solution is executed a number of times during evaluation, such as in regression problems.

Architecture

This layer is composed of three components which interact to examine the behaviour of protected method calls and to apply corrective measures in the event of their failure to prevent the program from crashing. These three components are:

Detection: Detects the occurrence of a specific type of fatal error within the affected method.

Diagnosis: Uses details of the error to determine its root cause within the source code of the affected method.

Repair: Uses the error diagnosis to apply corrective modifications to the source code of the affected method.

Instructions on how to detect, diagnose and repair specific types of errors are supplied to these components in the form of *error strategies*.

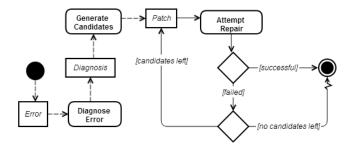


Figure 4: Architecture of the global robustness layer.

Immediately after an exception is thrown by a protected method, its details are passed to each of these error strategies, which try to detect if a certain type of error has occurred, the region of the source code responsible for producing the error, and how that code can be modified to remove the cause of the error.

After passing details of the error and method call to each error strategy, a set of candidate fixes are produced which are exhaustively tried until either the original error is no longer present, or there are no fixes left, in which case the original exception is thrown again, or a threshold number of tries is reached (10 in the work reported here).

Global Error Handling

The grammar as defined can produce two specific errors.

Missing Method Calls. Calls to non-existent methods are dealt with by changing all instances of the method name used to the name of an existing method that has with the lowest Levenshtein distance to the original.

Incorrect Number of Arguments. This is handled by reducing calls with too many parameters to the correct size, and padding calls with too few parameters with zeroes.

This transformation process manipulates the deepest method calls first (i.e. method calls that are used as parameters to other calls) to ensure that the resulting program is well-formed and that all calls to the affected method use the correct number of arguments.

Local Approach

Whereas the global robustness layer waits for a protected method to encounter a fatal error before attempting to repair the region of source code responsible for its cause, the local robustness layer is designed to achieve robustness by adapting the Ruby kernel itself to ensure that fatal exceptions are never raised in the first instance.

Instead of using a set of strategies to heal from specific errors, a set of changes are made to areas of the Ruby kernel capable of producing such errors, via a process known as "monkey patching" (Perrotta, 2010). By applying modifications to certain aspects of the programming language,

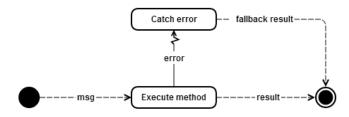


Figure 5: An overview of the local robustness layer design.

there is no need for a central entity to handle the execution and repair of protected methods; instead each fix acts independently, and the combination of them produces a thin robustness layer.

Local Error Handling

Missing Method Calls. To deal with calls to non-existent functions, the local approach implements Ruby's method_missing hook in the Module class, which is called when a requested method cannot be found inside a particular class.

The implementation of this function is similar to the approach taken to handling missing method calls in the global robustness layer; the function call is routed to the suitable function with the lowest Levenshtein distance to the requested function.

This re-routing mechanism also enforces a maximum distance constraint, which prevents the call from being routed to a function whose name is too far away from the requested method.

Incorrect Number of Arguments. Although the missing_method callback is capable of routing calls to non-existent functions towards their closest candidate function, this behaviour alone does not prevent an error being raised when the wrong number of arguments are then supplied.

To provide robustness against such errors, the implementation makes use of the method_added callback and the remove_method magic functions. Since there are no callbacks in place to handle function calls to existing functions with the incorrect number of arguments, the implementation instead captures and stores the object for every method added to the module in a private hash (indexed by the name of the method), via the method_added callback, before immediately removing the method from the module via the remove_method.

Every method call made to the module then invokes the missing_method callback, since methods are removed immediately after being added. The missing_method callback checks if the requested method "exists" by looking for its entry in the method hash; if there is no entry for the method then the closest candidate function is selected

Selection	Tournament
Mutation	Uniform Random
Crossover	Two Point
Replacement	Generational
Evaluation Limit	10,000

Table 1: EA setup used for all experiments.

instead (if this is not possible then an exception is raised), before adjusting its arguments to agree with the arity of the selected method.

The arguments provided with the method call are made to fit with the expected argument structure by removing excess arguments from the end of the arguments array or by padding the arguments with zeroes to the right to compensate for missing arguments.

Experiments

To determine the effect of the robustness layer on the evolution trajectory during GE, and to test the design hypothesis, we have conducted a series of benchmark evolutionary experiments. To perform these experiments, we used our own EA implementation, written in Ruby, whose components are detailed in Table 1. (After some preliminary testing, we found that two-point crossover yielded better results than the more traditional single point crossover.)

These experiments compare the performance and behaviour of GE using global, local and no robustness measures, across a number of benchmark multi-modal symbolic regression problems from McDermott et al. (2012), listed in Table 2.

The genome is a fixed-length list of 100 integers, each taking a value between 0 and $2^{31}-1$. The mutation operator uses uniform random replacement, replacing the value of a codon with a uniform random value from the legal range. Each genome is initialised with 100 uniform random values from the legal range.

The evaluation function measures the fitness of individuals as the sum of squared differences between their actual and expected results using data from a pre-generated training set. Hence low values are better than higher values, and the best achievable fitness is 0.

Calibration

Good parameter choices for each robustness measure, chosen from a given set of possible parameter values (Table 3), were determined through calibration based on the Keijizer-15 benchmark.

Due to limits upon time and computational resources, an exhaustive search of the space of all possible parameters is not possible. Instead, we employed the Relevance Estimation and Value Calibration of Evolutionary Algorithm

Name	Vars	Objective Function	Training Set
Koza-1	1	$x^4 + x^3 + x^2 + x$	U[-1, 1, 20]
Koza-3	1	$x^6 - 2x^4 + x^2$	U[-1, 1, 20]
Keijzer-12	2	$x^4 - x^3 + \frac{y^2}{2} - y$	U[-3, 3, 20]
Keijzer-14	2	$8/(2+x^2+y^2)$	U[-3,3,20]
Keijzer-15	2	$\frac{x^3}{5} + \frac{y^3}{2} - y - x$	U[-3,3,20]

Table 2: Regression problems used to perform comparison. U[a,b,c] is c uniform random samples drawn from a to b, inclusive, for the variable.

Parameter	Range
Tournament Size	[2, 10]
Elitism	[0.0, 0.5]
Mutation Rate	[0.001, 0.5]
Crossover Rate	[0.1, 1.0]
Population Size	[10, 200]

Table 3: The space of evolutionary parameter values.

Parameters (REVAC) method (Nannen and Eiben, 2007) to calibrate the evolutionary parameters of our algorithm.

We used the REVAC parameters of Smit and Eiben (2010), except with fewer evaluations (1000 instead of 5000) to reduce computational effort (Table 4). To measure the expected performance of each parameter vector, we use the mean best fitness (MBF) across a number of runs.

The results of calibration are shown in Table 5, and are the evolutionary parameters used for the evolutionary experiments.

Main Experiments

After determining good evolutionary parameters to use for each of the robustness measures, we compared the performance of each of the measures over 100 runs of each of the benchmark functions.

Results from these experiments (Figure 6 and Table 6) show an improvement in both the median and minimum best fitness values when using either local or global measures, compared to using no robustness measures, for all benchmarks except Koza-3.

In all cases, the use of local robustness measures gave a minimum best fitness that was at least as good as that obtained using global robustness measures, and was usually better; this may be due to the local robustness layer's ability to cope with an unlimited number of errors within any given program.

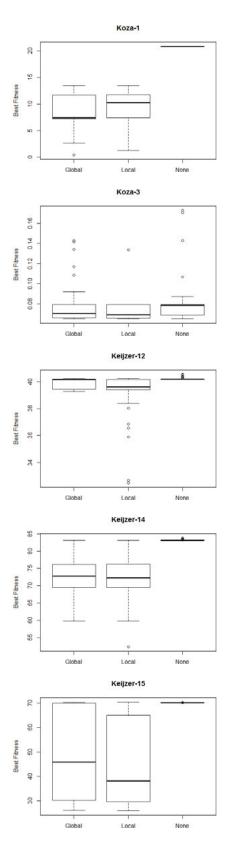


Figure 6: Distribution of best fitness values across 100 runs for each benchmark (low fitness values are better).

Population size	80
Best size	40
Smoothing coefficient	10
Repetitions per vector	10
Maximum number of vectors tested	1000

Table 4: REVAC parameters used during all tuning sessions.

Parameter	Global	Local	None
Tournament Size	2	4	2
Elitism	0.440	0.459	0.438
Mutation Rate	0.425	0.363	0.375
Crossover Rate	0.705	0.544	0.764
Population Size	161	161	189

Table 5: The parameter vectors for each measure.

Statistical Significance

We used the non-parametric Mann-Whitney U test to measure the statistical significance of the results, and to test the null hypothesis, H_0 : "the use of robustness measures have no effect on the fitness of the best individual found during evolution, compared to using no measures, when evolving a soft grammar."

We test to the standard 95% confidence level. We are performing 10 benchmark comparisons (Table 7), so we apply a Bonferroni correction and require each comparison to have a confidence level of 99.5%, or p < 0.005. Results given in Table 7 show that p < 0.005 on each benchmark except Koza-3. We reject the null hypothesis at the 95% confidence level, except for Koza-3.

Since using a large enough sample is usually enough to produce a statistically significant result, we also measure the effect size; we use the non-parametric Vargha-Delaney *A* value (Vargha and Delaney, 2000). From the results shown in Table 7, we can observe a large effect size between the use of global or local measures and no measures on almost of the benchmark functions, again except for Koza-3. These results suggest that the use of robustness measures produce a significant difference to the best fitness value found during evolution.

Conclusions

We have designed, implemented and tested two approaches that exploit reflection to perform self-repair in programs evolved using GE. We have demonstrated that both these measures have a statistically significant effect on the performance of GE using soft grammars in a series of benchmark functions.

Max U	Q Median	LQ	Min	
Global Measures				
3.415 11.68	2 7.387	7.291	0.460	
0.143 0.79	4 0.071	0.066	0.0652	
0.246 40.17	0 40.156	39.439	39.274	
3.195 76.04	9 72.756	69.552	59.833	
0.280 70.01	2 45.935	30.223	26.100	
Local Measures				
3.412 11.79	2 10.267	7.387	1.246	
0.134 0.07	9 0.069	0.066	0.065	
0.232 40.15	7 39.620	39.390	32.502	
3.195 76.20	8 72.318	69.541	52.359	
0.443 64.43	5 38.239	29.914	25.987	
No Measures				
0.801 20.80	1 20.801	20.801	20.801	
0.173 0.07	9 0.078	0.069	0.065	
0.537 40.18	5 40.185	40.158	40.156	
2 7 9 7 9 2 1 0	5 83.195	83.195	83.195	
3.787 83.19	3 63.193	05.175	05.175	
	3.415 11.68 0.143 0.79 0.246 40.17 3.195 76.04 0.280 70.01 3.412 11.79 0.134 0.07 0.232 40.15 3.195 76.20 0.443 64.43 0.801 20.80 0.173 0.07 0.537 40.18	Global Meas 3.415 11.682 7.387 0.143 0.794 0.071 0.246 40.170 40.156 3.195 76.049 72.756 0.280 70.012 45.935 Local Meass 3.412 11.792 10.267 0.134 0.079 0.069 0.232 40.157 39.620 3.195 76.208 72.318 0.443 64.435 38.239 No Measur 0.801 20.801 20.801 0.173 0.079 0.078 0.537 40.185 40.185	Global Measures 3.415 11.682 7.387 7.291 0.143 0.794 0.071 0.066 0.246 40.170 40.156 39.439 3.195 76.049 72.756 69.552 0.280 70.012 45.935 30.223 Local Measures 3.412 11.792 10.267 7.387 0.134 0.079 0.069 0.066 0.232 40.157 39.620 39.390 3.195 76.208 72.318 69.541 0.443 64.435 38.239 29.914 No Measures 0.801 20.801 20.801 20.801 0.173 0.079 0.078 0.069 0.537 40.185 40.185 40.158	

Table 6: Results from the benchmark experiments (low fitness values are better)

Measure	Benchmark	p	A
Global	Koza-1	3.293×10^{-30}	1.000
	Koza-3	0.0173	0.595
	Keijzer-12	3.761×10^{-12}	0.781
	Keijzer-14	1.003×10^{-35}	0.982
	Keijzer-15	1.304×10^{-19}	0.864
Local	Koza-1	4.163×10^{-39}	1.000
	Koza-3	0.0011	0.630
	Keijzer-12	4.483×10^{-20}	0.872
	Keijzer-14	2.670×10^{-37}	0.995
	Keijzer-15	3.044×10^{-30}	0.962

Table 7: Statistical significance (Mann-Whitney U test p) and effect size (Vargha-Delaney A value) of use of global and local robustness measures each compared to no measures. An A value > 0.56 is a "small" effect; an A value > 0.71 is a "large" effect.

Limitations

Although the local robustness layer is capable of handling the errors investigated here, to handle each error, a series of changes to the Ruby kernel has to be made. This approach is thus limited by the ability of the Ruby kernel to catch such errors. More complex errors, such as name errors raised when an unreferenced variable is used, require the robustness layer to know their context and to modify the original source code, and cannot be intercepted and dealt with on the fly. Since the global robustness layer is capable of examining the context of errors, and has the required ability to apply changes to the source code, it is better equipped to diagnose and repair a far wider category of errors.

Whilst the local robustness measures performed well across all of the problems, an unfortunate consequence of their changes to the Ruby kernel restricts them to a single thread during evolution. However, this shortcoming may be addressed using Ruby's recently introduced *refinement* functionality (Cooper, 2010), which allows kernel changes to be localised within a given context.

Another weakness of the local approach is that its inherent handling of errors on the fly means that the semantics of programs produced during evolution may not be the same when the local robustness layer is not deployed, since the code is never changed and errors will no longer be dealt with. One solution to this problem may be to pass individuals to a finaliser at the end of the evolutionary process, which analyses the execution of their associated program and monitors exceptions caught by each of the patches to determine the effective source code of the program.

Future Work

Having demonstrated the effectiveness of robustness measures when combined with soft grammars, we are now investigating the use of radically more expressive grammars, modelling subsets of an entire programming language, capable of universal computation.

We believe that by taking this approach, we can free the programmer from performing arbitrary decisions on their choice of grammar, and that entirely novel solutions to problems, beyond the scope of conventional hard grammars, may be discovered.

We are also investigating the use of techniques which allow the use of alternative meta-languages to perform correction, instead of the same language as the program, allowing programs to be evolved in arbitrary languages, regardless of their support for reflection.

Our long term goal is to use such techniques in the development of an open-ended an open-ended reflective software architecture to support open-ended evolution *in silico* (Stepney and Hoverd, 2011).

References

- Cooper, P. (2010). Ruby Refinements: An Overview of a New Proposed Ruby Feature. http://www.rubyinside.com/rubyrefinements-an-overview-of-a-new-proposed-ruby-feature-3978.html.
- Dempsey, I., O'Neill, M., and Brabazon, A. (2009). Foundations in Grammatical Evolution for Dynamic Environments. Springer.
- Ferber, J. (1989). Computational reflection in class based objectoriented languages. In OOPSLA '89, pages 317–326. ACM.
- Haydarlou, A. R., Overeinder, B. J., and Brazier, F. M. T. (2005).
 A self-healing approach for object-oriented applications. In Proc. 3rd International Workshop on Self-Adaptive and Autonomic Computing Systems, pages 191–195.
- Kephart, J. and Chess, D. (2003). The vision of autonomic computing. *Computer*, 36(1):41–50.
- Maes, P. (1987). Concepts and experiments in computational reflection. ACM Sigplan Notices, 22(12):147–155.
- McDermott, J., White, D. R., Luke, S., Manzoni, L., Castelli, M., Vanneschi, L., Jaskowski, W., Krawiec, K., Harper, R., De Jong, K., and O'Reilly, U.-M. (2012). Genetic programming needs better benchmarks. In GECCO '12, pages 791–798. ACM.
- Nannen, V. and Eiben, A. E. (2007). Relevance estimation and value calibration of evolutionary algorithm parameters. In *IJCAI'07*, pages 975–980. AAAI Press.
- O'Neill, M. and Brabazon, A. (2006). Grammatical Swarm: The generation of programs by social programming. *Natural Computing*, 5(4):443–462.
- O'Neill, M., Ryan, C., and Nicolau, M. (2001). Grammar defined introns: An investigation into grammars, introns, and bias in grammatical evolution. In *GECCO 2001*, pages 97–103.
- Perrotta, P. (2010). Metaprogramming Ruby. Pragmatic Bookshelf.
- Ryan, C., Collins, J., , and O'Neill, M. (1998). Grammatical Evolution: Evolving programs for an arbitrary language. In *Proc.*1st European Workshop on Genetic Programming, volume 1391 of LNCS, pages 83–95. Springer.
- Smit, S. K. and Eiben, A. (2010). Beating the 'world champion' evolutionary algorithm via REVAC tuning. In *CEC 2010*, pages 1–8. IEEE.
- Smith, B. C. (1982). *Reflection and semantics in a procedural language*. PhD thesis, Massachusetts Institute of Technology, Laboratory for Computer Science.
- Stepney, S. and Hoverd, T. (2011). Reflecting on open-ended evolution. In *ECAL 2011, Paris, France, August 2011*, pages 781–788. MIT Press.
- Stump, A. (2009). Directly reflective meta-programming. *Higher Order Symbol. Comput.*, 22(2):115–144.
- Tschudin, C. F. (2003). Fraglets a metabolistic execution model for communication protocols. In *Proc. 2nd Annual Symposium on Autonomous Intelligent Networks and Systems (AINS), Menlo Park.*
- Vargha, A. and Delaney, H. D. (2000). A critique and improvement of the CL common language effect size statistics of McGraw and Wong. J. Educ. Behav. Stat., 25(2):101–132.

Studying the Evolvability of Self-Encoding Genotype-Phenotype Maps

Andrew M. Webb and Joshua Knowles

School of Computer Science, University of Manchester, UK andrew.webb@manchester.ac.uk

Abstract

We introduce a model of reproduction in which the genotypephenotype (G-P) map is able to evolve. In this model, Each organism implements a G-P map, determining how the organism is encoded in its genome. Crucially, it also determines how the G-P map itself is encoded. We call these maps 'self-encoding'. We relate this model to recent artificial life research, and back to the seminal work of John von Neumann. We simulate populations of organisms that have as their genome and G-P map the axiom and production rules of an L-system. The populations are given the task of optimizing a dynamic fitness function. Our purpose is to study whether the self-encoding property has any effect on the evolution of evolvability, and to look for other factors that lead to the evolution of G-P maps that confer evolvability. We find that evolvability does evolve, but only when we add constraints to the model.

Introduction

Our principal reason for studying the genotype-phenotype (G-P) map is its role in determining evolvability. We use Hansen's definition (Hansen, 2006):

"Evolvability is the ability of the genetic system to produce and maintain potentially adaptive genetic variants."

Higher evolvability means a greater probability of introducing adaptive variants. It is a variational property, related to the way that variations are introduced into the phenome.

Mutations in the genome are more or less random¹. The genome is an encoded form of the phenome, with the G-P map implementing the decoding function. It can potentially translate the random mutations in genome into directed changes in the phenome (Wagner and Altenberg, 1996; Hansen, 2006). For example, the G-P map might perform error correction when determining one attribute of the phenome, making it less variable than other attributes. Or it might use a single gene to determine two attributes, ensuring

that the two can only vary together. We will say that the G-P map *mediates* variation in the phenome, in the sense that it acts as a mediator between the random source of variation and variation that actually occurs.

A G-P map increases evolvability if it translates random mutations into suitably directed (i.e., sometimes adaptive) changes. Life on Earth is believed to have evolved towards higher evolvability through changes to the G-P map (Dawkins, 2003; Pigliucci, 2008).

When and how does evolvability evolve? In what kinds of environment does it happen? How can beneficial G-P maps be selected for when they only confer a future advantage and when selection can only act on current fitness? Recent work in artificial life has aimed to answer such questions by studying model organisms in which the G-P map can evolve. We introduce our own model of reproduction, in which the G-P map is 'self-encoding'; it determines how the G-P map itself is encoded in the genome.

The purpose of our research is to learn what factors lead to the evolution of G-P maps that confer greater evolvability. Our contributions are as follows.

- We describe and compare recent artificial life research in which model organisms within the Avida platform, or an extension of it, are studied with the purpose of understanding the evolution of the G-P map.
- We suggest potential problems with these models, and with using Avida to study the evolution of the G-P map in general.
- We introduce a model of self-reproduction in which organisms implement a 'self-encoding' G-P map.
- We simulate populations of these organisms, using L-system production rules as their G-P map, that evolve to optimize a dynamic fitness function. We find that evolvability fails to evolve unless we add constraints to the model.

Context and Related Work

In this section, we describe two models of self-reproduction used in recent artificial life research to study the evolution

¹There are conflicting definitions of 'genotype'. Here we use 'genome' to refer to the total genetic information of an organism, and 'phenome' to refer to the set of all traits of an organism. We use the established term 'genotype-phenotype map' to refer to the mapping between them.

of the G-P map. Since both models are based on the Avida platform, we also describe the default reproduction mechanism in Avida. For comparison, we describe the reproduction mechanism of John von Neumann's *universal constructor*.

Von Neumann's Universal Constructor

John von Neumann developed a two-dimensional, 29-state cellular automaton, and a structure within it that he called the *universal constructor* (Von Neumann and Burks, 1966). The universal constructor has, roughly, the following components.

- A one-dimensional 'tape' **G**, which in inactive; on its own, it doesn't do anything.
- An active 'machine' **P**, which consists of the following four components.
 - A 'decoder' A, which implements a decoding function.
 It reads the contents of the tape G, decodes the contents into the specification of a two-dimensional structure, and then builds that structure elsewhere in the cellular automaton.
 - The 'tape copier' B, which builds a copy of the tape G adjacent to the structure built by the decoder.
 - The 'coordinator' **C**, which coordinates the actions of the decoder and tape copier.
 - The 'ancillary machine' D, which can perform any arbitrary action as long as it doesn't interfere with the other components.

If the tape ${\bf G}$ contains an encoded description of ${\bf P}$, relative to the decoding function implemented by ${\bf A}$, then the pair constitute a self-reproducing machine; the part ${\bf B}$ constructs a copy of ${\bf G}$, and the part ${\bf A}$ decodes ${\bf G}$ to construct ${\bf P}$. In order to be a self-reproducer, the machine has to do two things. First, it has to implement a mechanism for copying the tape. Second, it has to implement a decoding mechanism that decodes the tape into a description of the tape-copying and decoding mechanisms.

This is a convincing model of biological reproduction that includes a G-P map. The analogy to life is as follows. **G** is the genome of the organism. **P** is the phenome. **A** is the G-P map, encompassing the genetic code and development processes. **B** is the mechanism for copying the genome. **D** is any behaviour of the organism not directly related to reproduction.

Mutations in **G** will manifest as a change in **P** in the offspring. Von Neumann noted that if the mutation affects only the ancillary machine, then the offspring will still be a self-reproducer. He believed that any mutation affecting the decoder would render the offspring infertile, as the decoding function it implements would no longer complement the encoding of **G**. A change to the decoder that leaves the offspring fertile corresponds to a change in the G-P map.

When G is mutated, how this manifests as a change in P of the offspring is mediated by the part A implemented

by the parent. This includes changes *to* part **A**; it mediates all variations in the offspring. Barry McMullin has been a proponent of studying this model of reproduction (McMullin, 2012).

Avida

Avida is a popular software platform for studying self-replication (Ofria and Wilke, 2004), in which each organism is a program in an assembly-like language. Avida has a cellular structure; each organism has its own private memory, and the organisms are arranged in a grid. The population of Avidans execute their instructions in parallel.

Avida simulations are initialized with a single, hand-designed self-replicating program called the *ancestor*. The default ancestor replicates by setting up and iterating over a copy loop, in which the 'copy' instruction is repeatedly executed to read from and write to successive memory locations. Once it has copied the entire contents of its memory, it executes the special *divide* instruction. This places a new child Avidan in a neighbouring cell, giving it the recently copied instructions.

Avidans can perform additional tasks as well as self-replication, and the experimenter can adjust the 'metabolic rate' of an organism (the rate at which it executes instructions) based on its performance on these additional tasks.

There are multiple types of mutation in Avida. The 'copy' instruction has a low probability of writing the wrong instruction, and 'cosmic ray' mutations can change any instruction in memory at any time.

An Avidan's sequence of instructions, being the hereditary information that is transmitted between generations, is its genome. The phenome of an Avidan is its behaviour when run. Since the genome determines the phenome, the G-P map is fixed.

As McMullin notes, in von Neumann's terminology, the **G** and **P** parts of an Avidan are the same, and consist only of components **B** and **D**. Avidans don't reproduce by applying a decoding function to an encoded description of themselves; they directly read and copy the contents of their own memory.

Implementing the 'von Neumann architecture' in Avida

Hasegawa and McMullin study a self-reproducing ancestor program based on von Neumann's model of reproduction (Hasegawa and McMullin, 2012). One part of the program, identified with **G**, is not executed but is read as data. The other part, identified with **P**, is executed, and does the following.

- 1. Enter a copy loop, identified with the tape copier **B**. Unlike the copy loop of the default Avidan ancestor, which copies the entire contents of memory, it only copies part **G**.
- 2. Apply a decoding function, identified with the decoder **A**, to **G**, and write the result to the region of memory adjacent

to the copy of G.

3. Execute the divide instruction.

During this process, mutations can happen in any part of the memory. Like von Neumann's constructor, Hasegawa and McMullin's ancestor has the property that when the part **G** is mutated, the change observed in **P** of the offspring is mediated by the decoding function implemented by the parent. However, it is possible for 'mutations' to happen directly in the part **P**. These mutations will cause changes to **P** that are unmediated by the decoding function.

One of their results is that populations quickly reverted to the self-inspection form of reproduction more commonly seen in Avida, without a decoding step.

Evolving the 'hardware' in Avida

Egri-Nagy and Nehaniv implemented a variant of Avida, in which each organism has its own set of data structures and its own high-level programming language, with instructions composed of Avida's lower-level instructions (Egri-Nagy and Nehaniv, 2003). When an organism is born, the first part of its memory is read as a specification of that organism's data structures and instruction set (its 'hardware'). The organism then starts executing instructions as normal.

They refer to the hardware specification as the 'G-P map', because it changes the result of executing the program (the 'genome'). However, unlike in Hasegawa and McMullin's work, the genome doesn't contain an encoded form of the G-P map; the mechanism of reproduction, like that of the default Avida ancestor, is to enter a copy loop that copies the entire contents of memory to the offspring, introducing copy errors.

The part labelled the 'G-P map' mediates the changes in the behaviour of the organism as a result of mutations in the genome. However, the G-P map itself is subject to random mutations, and the encoding of this part of the organism is fixed; variation in the G-P map is unmediated.

General Limitations

We suggest that Avida may not be an ideal platform for studying the evolution of G-P maps for the following reasons.

- Because it was designed to allow reproduction by self-inspection and copying, without a decoding step, this method of reproduction will inevitably be the most efficient in Avida. One of the results of Egri-Nagy and McMullin's work is that simulations initialized with their ancestor quickly become dominated by 'self-inspection' reproducers.
- The assembly-like programming language used by Avida isn't designed for easily expressing developmental processes.

Two of the types of mutation in Avida can act on any location in memory; for any Avidan that has parts equivalent to G and P, some mutations will happen directly in P, unmediated by the decoder.

Self-Encoding Genotype-Phenotype Maps

Here we introduce a model of reproduction in which each organism implements a G-P map, which determines how the organism is encoded in its genome. Moreover, it determines how the G-P map *itself* is encoded in the genome. For brevity, in the following we use 'genotype-phenotype map' and 'decoder' synonymously. We call the G-P maps 'self-encoding', and call the organisms 'self encoders'. The model differs from previous work in the following ways.

- The organisms do not implement a mechanism for copying their genome. In von Neumann's terminology, our organisms are missing part B. The reason is that we want to focus on part A, the decoder, and we know precisely what the behaviour of part B should be. The genome-copying step is built into the model.
- We initialize the population with randomly generated organisms, rather than hand-designed self-reproducing organisms, in order to avoid accidentally initializing the population in an evolutionary dead-end.
- Only the genome is mutated. All variation in the phenome is mediated by the decoder, including variation in the decoder itself.
- Rather than an assembly programming language, our model will use decoding mechanisms that are biologically inspired.

With the copying of the genome built into the model, and the initial decoder population initialized randomly, the initial state is analogous to a population of self-replicating RNA. The random initial 'decoders' are like active molecules that interact with the RNA to produce other molecules. The initial decoders are not self-reproducing; the emergence of self-reproducing decoders corresponds to the emergence of a developmental step on top of an existing replication mechanism.

Each organism in the model consists of a genome and a phenome. The phenome consists of a decoder, and another part that we will apply a fitness function to and adjust the metabolism of the organism based on the result. We call this part the *solution*. See Figure 1.

The population lives on a discrete grid, and is updated iteratively, in parallel, by looping over the grid. Algorithm 1 shows one such iteration. Fitter organisms are rewarded by 'stealing' update cycles from less fit neighbours. When an organism creates a new child, the child receives a mutated copy of the parent's genome. This genome becomes the starting point for the 'embryo', which is a working copy of the

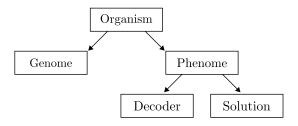


Figure 1: The structure of an organism in our model. Arrows indicate a 'contains' relationship.

data operated on by the parent's decoder. Each time a parent organism updates, it goes through one decoding step, applying its decoder to the embryo of its child. Later in the paper we give an example of a decoder being applied iteratively. When decoding is complete, the embryo is interpreted as the specification of a decoder and a solution. These are given to the child, and the solution is used to set the child's fitness. The child is then placed somewhere in the neighbourhood of the parent.

Algorithm 1 One time step of the simulation.

```
1: for each cell in grid do
        organism p \leftarrow fitter of organism in cell and organism
2:
    randomly selected from 3 \times 3 neighbourhood
 3:
        if p has no embryonic child then
4:
             p.child.genome \leftarrow mutated\_copy(p.genome)
             p.child.embryo \leftarrow p.child.genome
 5:
 6:
        p.child.embryo \leftarrow p.decoder(p.child.embryo)
        if decoding is complete then
 7:
             (d, s) \leftarrow interpret(p.child.embryo)
 8:
9:
             p.child.decoder \leftarrow d
10:
             p.child.solution \leftarrow s
11:
             p.child.fitness \leftarrow fitness\_function(s)
             place child on grid in 3 \times 3 neighbourhood of p
12:
```

Selection is partly implicit, partly explicit; there is an implicit selection pressure to be an efficient and robust reproducer, and we explicitly apply selection pressures by rewarding organisms for the contents of their solutions.

The Decoding Mechanism

We use L-system production rules as the decoding mechanism. L-systems were first used to model plant development (Lindenmayer, 1968), but have since become popular as a more general model of development. An L-system consists of an alphabet, an axiom, which is a string in that alphabet, and a set of production rules. Each production rule specifies how to rewrite a substring to another substring. L-systems grow strings iteratively, starting from the axiom, by applying as many production rules as possible in parallel.

For example, an L-system might have the axiom

Α

and the production rules

$$\begin{array}{l} {\rm A} \rightarrow {\rm BA}, \\ {\rm B} \rightarrow {\rm A} \end{array}$$

The first few strings generated by this L-system are

A BA ABA BAABA ABABAABA

The L-systems we use are context sensitive (Prusinkiewicz et al., 1990), so the left-hand side of each production rule contains three symbols: the left context, the symbol to rewrite, and the right context. For example, the production rule

replaces the symbol \mathbb{B} , if it has an \mathbb{A} on the left and a \mathbb{C} on the right, with the symbols \mathbb{DEF} . We use context-sensitive L-systems because they allow recursive application of production rules, while also making it possible for the process of applying rules to terminate rather than recursing indefinitely.

Each organism has a set of production rules as its decoder, and its genome is used as the axiom of the L-system. Every L-system has the same alphabet of 10 symbols. The axiom of a parent is copied to its child and mutated. Starting with the child's axiom, the parent's production rules iteratively grow a string. One decoding step consists of applying as many production rules as possible in parallel.

When there are no more rules to apply, decoding is complete. The resulting string specifies both the production rules and the solution of the child. The symbol sequence AA is a punctuation mark separating the two parts. For example, the organism with the production rule

$$A < B > A \rightarrow DEFAADEF$$

operating on the axiom

ABA

will produce the string

ADEFAADEFA

which the AA punctuation separates into the strings

ADEF,
DEFA

with the first string specifying the production rules of the offspring, and the second string specifying the solution.

Since all production rules have three symbols on the left-hand-side and an arbitrary number of symbols on the right-hand-side, the production rules are specified in the string as follows. The first three symbols specify the left-hand-side of a rule. Now there is another kind of punctuation: each of the sequences {BB, BC, BD, CB, CC, CD, DB, DC, DD} indicate the end of the current rule; the right-hand-side of the rule consists of all of the symbols up to, but not including, the next such punctuation. The next rule is then built from the symbols following that punctuation. For example, the string

DEFDEFBCGHIGHI

contains the punctuation sequence ${\tt BC}$, and so specifies the two production rules

$$\begin{array}{l} {\rm D}{<}{\rm E}{>}{\rm F} \rightarrow {\rm DEF}, \\ {\rm G}{<}{\rm H}{>}{\rm I} \rightarrow {\rm GHI} \end{array}$$

The use of sequences of symbols as punctuation precludes their use as data; it is impossible to build a rule that explicitly rewrites to any substring containing a punctuation sequence, though it can be done indirectly.

The Fitness Function

We adjust the metabolism of the organisms based on the result of applying a fitness function to their solutions. Since L-systems are well suited to growing self-similar patterns, we use a fitness function that rewards strings with extensive repetition, in the hope that decoders well suited to the problem will emerge. The fitness function counts the number of occurrences of the symbol $\mathbb E$ in the solution, up to a maximum of 150.

Since we are looking for the evolution of evolvability, we use a fitness function that changes periodically in a structured way; if the fitness function was static, then once the population performed perfectly there would be no meaningful sense in which it could be evolvable or not. After a while we switch to rewarding for occurrences of the F symbol, then G, H, I, and back to E.

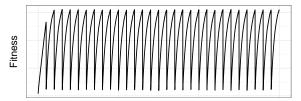
Simulations and Results

We ran simulations in order to find factors leading to the evolution of evolvability. We were also looking for the emergence of stable, self-reproducing decoders: organisms whose production rules, applied to their own axiom string, yield a description of themselves.

Since the fitness function changes periodically, the average fitness of a population over time will look like Figure 2a, increasing until the change point and then dropping to almost zero. We switch the fitness function every 100,000 fitness function evaluations. We measure the average fitness within each window of 100,000 evaluations, giving a single number measuring how effective the population was at maximizing the fitness function within that window.

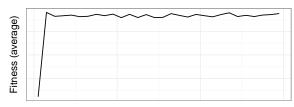
Figure 2b shows an example of the window-averaged fitness over time. Within a given window, the height of this graph gives an indication of the evolvability of the population with respect to the fitness function in that window. A positive slope would indicate the evolution of evolvability.

We compared the performance of a population of self encoders, using the context-sensitive L-system decoding mechanism, with two other methods of optimizing the fitness function. One is a simple genetic algorithm, in which there are no production rules; the genome string directly represents a solution to the fitness function. We use the genetic algorithm



Time (# fitness function evaluations)

(a) An example of average population fitness over time.



Time (# fitness function evaluations)

(b) An example of the fitness over time, averaged over each fitness function window. Within each window of 100,000 evaluations we use this single number as an indication of evolvability.

as a baseline against which we compare the performance of the other methods.

The other method used for comparison does have a set of production rules, but these are used to construct only a solution, rather than a solution and the production rules of the next generation; at each reproduction step, the genome string is copied to the offspring and mutated, the parent's production rules are applied to the genome string, and the result specifies the solution. The parent's production rules are then also copied to the offspring and mutated. We call this the *solution encoder*, since the organisms control only how they encode the solution, and the encoding of the decoder is fixed. The purpose of comparing against the solution encoder is to determine whether the self-encoding property has an effect on the evolution of evolvability.

Each simulation ran for 3,000,000 function evaluations/births. We ran each simulation forty times for each method. In the fitness plots in the following sections, the solid or dashed line shows the median fitness amongst these runs over time, and the surrounding shaded region lies between the 25th and 75th percentiles.

Simulation 1

In the first simulation, we compared a population of self encoders, solution encoders, and the genetic algorithm, each with a population size of 20×20 . Figures 3 and 4 show the results.

The solution encoder performs well. The performance profile of the self encoder population is similar to, if a little worse than, that of the genetic algorithm. The reason is that the population is quickly dominated by organisms with empty sets of production rules. The following are some example

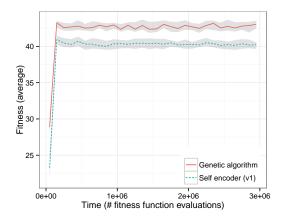


Figure 3: The window-averaged fitness of the self encoder and genetic algorithm over time in simulation 1, with a population size of 20×20 .

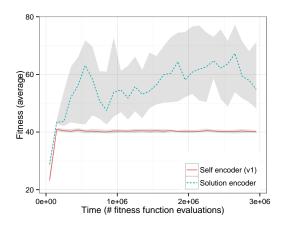


Figure 4: The window-averaged fitness of the self encoder and solution encoder over time in simulation 1, with a population size of 20×20 .

genomes from the self encoder population.

AAGGGHGGDGGGJGG AAAGEEEEEEAEEEEFE GAAJGJFJJJEJHJJJJJJ

The organisms are evolving to have AA, the punctuation that indicates the end of the production rule specification and the start of the solution, right at the start of their genome strings; they will have no production rules, nor will any of their descendants, unless mutation disrupts the punctuation.

We suspect that this happened because the initial search over production rule sets is detrimental, and so selection acts against any organism with production rules. To test this, we introduced a variation of the self encoder, in which AA only acts as punctuation if it appears in the second half of the output string. The result is to force the self encoders to have non-empty production rule sets. We call the new population

the self encoder (v2), and the original the self encoder (v1).

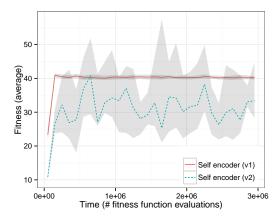


Figure 5: The window-averaged fitness of the self encoder (v1) and self encoder (v2) over time in simulation 1, with a population size of 20×20 .

The typical organism in the self encoder (v2) population has a single, long production rule in its decoder. We observed no instances of single-generation self-reproducing organisms, but we frequently found examples of two-generation reproducers, in which an organism and its 'grandchild' have the same production rules. Gestation times were low; typically, organisms went through one or two decoding steps. We found that, often, the genome explicitly contained the solution.

Figure 5 compares the performance of a population of self encoders (v1) and self encoders (v2). The population of self encoders (v2) doesn't perform especially well, but the wide inter-percentile range suggests that, sometimes, the population evolves decoders that confer evolvability.

Simulation 2

The high variance in the evolvability of the self encoder (v2) population in simulation 1 suggests that, sometimes, good sets of production rules do evolve. This motivated us to increase the population size to 20×100 for the next simulation. Since direct competition is between neighbours, the increased population size and rectangular shape has the effect of reducing the rate at which the descendants of a fitter organism spread in the population. This ought to allow multiple competing decoders to co-exist in the population at any time, allowing greater exploration over decoders before any one of them dominates the population.

Figures 6 and 7 show the results. The self encoder (v1) population is again dominated by organisms with empty sets of production rules. The self encoder (v2) performs worse than the self encoder (v1) for the first six or so fitness function windows, then performs better. This appears to show long-term adaptive evolution in the G-P map, the beneficial effect of which remains when the fitness function changes.

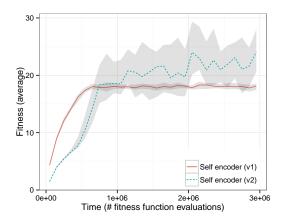


Figure 6: The window-averaged fitness of the self encoder (v1) and self encoder (v2) over time in simulation 2, with a population size of 20×100 .

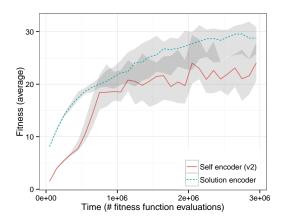


Figure 7: The window-averaged fitness of the self encoder (v2) and solution encoder over time in simulation 2, with a population size of 20×100 .

Simulation 3

In the previous two simulations, the self encoder (v2) population became quickly dominated by organisms with decoders consisting of one long production rule. The organisms are not evolving to make use of the 'end rule' punctuation. We measured the relative frequencies of the symbols in the decoded strings over a whole run as follows.

ſ	A	В	С	D	E
ſ	0.16	0.04	0.04	0.04	0.14
ſ	F	G	Н	I	J
ĺ	0.12	0.14	0.14	0.12	0.06

Symbols E-I appear frequently because they are each, in turn, rewarded for. Symbol A appears frequently because it appears in the punctuation that indicates the start of the solution, without which the solution would be empty. Symbols B, C, and D, which appear in the rule-separation punctua-

tion, appear less frequently than symbol J, which doesn't serve any purpose; they are actively selected against. To see what would happen if organisms evolved to have large sets of shorter production rules, we added a restriction: the right-hand-side of each production rule must be less than or equal to ten symbols long.

Figures 8 and 9 show the results of this simulation. Now the self encoder (v2) out-performs the self encoder (v1) almost immediately, showing the adaptive value of searching over decoders. But the self encoder (v1) population is still dominated by organisms that evolve empty sets of production rules. Without the constraint to force non-empty production rule sets, the initial detrimental effect of searching over decoders still prevents the search from ever beginning.

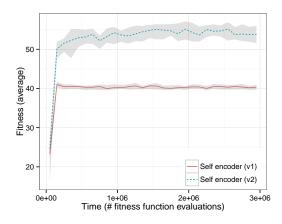


Figure 8: The window-averaged fitness of the self encoder (v1) and self encoder (v2) over time in simulation 3, with a population size of 20×20 and a limit on the size of each production rule.

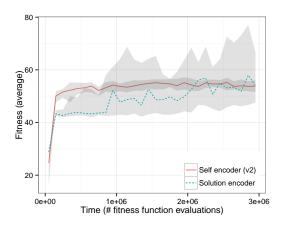


Figure 9: The window-averaged fitness of the self encoder (v2) and solution encoder over time in simulation 3, with a population size of 20×20 and a limit on the size of each production rule.

In the self encoder (v2) population, gestation times were higher than in simulation 2, with organisms typically going through four or five decoding steps. We found that, more frequently than in simulation 2, the solution wasn't contained explicitly within the genome, but was actually grown by applying the production rules. We observed a kind of 'fuzzy' reproduction, in which an organism is similar, but not identical, to its parent or grandparent. For example, the following two sets of production rules are from one organism (left) and its grandchild (right).

```
B < E > G \rightarrow AGGAGGAGGA
                                          B<E>G → AGGAGAGEGA
A < G > B \rightarrow EGAGGAGGAG
                                          A < G > B \rightarrow EGAGGAGGGA
A < G > G \rightarrow AGGAGGAGGA
                                          A < G > G \rightarrow GAGAGABEJF
\texttt{D}{<}\texttt{F}{>}\texttt{H} \to \texttt{AGGAGGAGGA}
                                          \texttt{D}{<}\texttt{F}{>}\texttt{H} \to \texttt{BEFEIFHBEF}
E < A > E \rightarrow IFHAAGGAGG
                                          E < C > C \rightarrow FDACJEGDAC
\texttt{E}{<}\texttt{G}{>}\texttt{A} \to \texttt{GGAGGAGGGH}
                                          E < G > A \rightarrow GGAGGGAGGG
F < E > B \rightarrow EABEAEBEAB
                                          E < G > G \rightarrow AGGGAGGAGA
\texttt{F}\texttt{<}\texttt{E}\texttt{>}\texttt{F}\to\texttt{EBEABEAEFE}
                                          F < E > F \rightarrow FHJDGAGEGA
\mathsf{G}{<}\mathsf{A}{>}\mathsf{G} \to \mathsf{GAGGAGGGAG}
                                          G < A > G \rightarrow EGAGGAGGGG
                                          G<E>G → AGGAGGGGAG
G < G > A \rightarrow GBEGAGGAGG
                                          G < G > A \rightarrow GGGAGAGGAG
G < G > G \rightarrow AGGJFE
                                          G < G > G \rightarrow AGGAGBEGAG
```

Conclusions and Future Work

We have described a model of reproduction, called the self encoder, in which the genome contains an encoded description of the G-P map. In this model, all variation in the offspring, including variation in the G-P map itself, is mediated by the G-P map implemented by the parent. We simulated populations of organisms whose G-P maps were context-sensitive L-systems. Our purpose was to discover whether the self-encoding property has any effect on the evolution of evolvability, as well as to find other factors that lead to the evolution of G-P maps that confer evolvability (with respect to a dynamic fitness function) on the organisms that implement them.

We also looked for the emergence of self-reproducing G-P maps. We observed a kind of 'fuzzy' reproduction, in which organisms had similar G-P maps to their grandchildren or great-grandchildren. In future work we aim to measure the degree of similarity between generations.

We found no conclusive evidence of a difference between the ability of organisms with and without the self-encoding property to evolve evolvability, nor did we rule out such a difference. We found that in the self-encoding population 'good' G-P maps failed to evolve until we added constraints to the model to push evolution towards them. However, once we added the constraints good G-P maps evolved quickly. This was to be expected, as evolvability confers a future advantage, whereas selection acts based on current fitness.

First, we found that populations became dominated by organisms whose L-systems had no production rules. Once we added the constraint that forced them to have non-empty sets of rules, in some cases the population went on to discover

G-P maps that increased evolvability. Second, increasing the population size led to the evolution of good G-P maps, since it allowed multiple G-P maps to co-exist for longer; it allowed the exploration of and selection between G-P maps. Third, we found that organisms tended to evolve to have a single long production rule. We added a constraint to force the population to evolve to have a large number of short production rules, and found that as a result the population discovered better G-P maps.

In future work we aim to study a wider range of constraints and mechanisms causing selection for good G-P maps, with an emphasis on mechanisms that are hypothesized to operate in nature.

References

- Dawkins, R. (2003). The evolution of evolvability. *On Growth, Form and Computers*, pages 239–255.
- Egri-Nagy, A. and Nehaniv, C. L. (2003). Evolvability of the genotype-phenotype relation in populations of self-replicating digital organisms in a Tierra-like system. In *Advances in Artificial Life*, pages 238–247. Springer.
- Hansen, T. F. (2006). The evolution of genetic architecture. Annual Review of Ecology, Evolution, and Systematics, pages 123– 157.
- Hasegawa, T. and McMullin, B. (2012). Degeneration of a von Neumann self-reproducer into a self-copier within the Avida world. In *From Animals to Animats* 12, pages 230–239. Springer.
- Hasegawa, T. and McMullin, B. (2013). Exploring the point-mutation space of a von Neumann self-reproducer within the Avida world. In *Advances in Artificial Life, ECAL*, volume 12, pages 316–323.
- Lindenmayer, A. (1968). Mathematical models for cellular interactions in development i. filaments with one-sided inputs. *Journal of theoretical biology*, 18(3):280–299.
- McMullin, B. (2012). Architectures for self-reproduction: Abstractions, realisations and a research program. In *Artificial Life*, volume 13, pages 83–90.
- Ofria, C. and Wilke, C. O. (2004). Avida: A software platform for research in computational evolutionary biology. *Artificial life*, 10(2):191–229.
- Pigliucci, M. (2008). Is evolvability evolvable? *Nature Reviews Genetics*, 9(1):75–82.
- Prusinkiewicz, P., Lindenmayer, A., and Hanan, J. (1990). The algorithmic beauty of plants. *The virtual laboratory (USA)*.
- Ray, T. S. (1991). An approach to the synthesis of life.
- Stanley, K. O. and Miikkulainen, R. (2003). A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130.
- Von Neumann, J. and Burks, A. W. (1966). *Theory of self-reproducing automata*. University of Illinois Press, Urbana.
- Wagner, G. P. and Altenberg, L. (1996). Complex adaptations and the evolution of evolvability. *Evolution*, 50(3):967–976.

Ecosystem memory is emergent from local-level natural selection

Daniel A. Power¹, Eörs Szathmáry² and Richard A. Watson³

¹Institute for Complex Systems Simulation, University of Southampton, U.K.

²Center for the Conceptual Foundations of Science, The Parmenides Foundation, Germany

³Natural Systems Group, University of Southampton, U.K.

dap1e12@soton.ac.uk

Because the form of an ecosystem is shaped chiefly through the selection and amplification of chance genetic and environmental events at lower levels of organisation (sensu Maynard Smith and Szathmáry (1997)), the number of evolutionary outcomes for these systems is enormous. Theoretical models of ecosystem evolution and function generally show sensitivity to initial conditions and small disturbances that result in very different behaviours for mature systems (May, 2001). These non-linearities mean that ecosystem function is historically contingent; we observe path dependency (over evolutionary timescales) as well as those non-linearities (including hysteresis) that occur over ecological timescales. For researchers seeking to understand the degree to which ecosystem properties are the result of abiotic environmental conditions and the extent to which they are emergent from self-organisation (Levin, 1998), this contingency adds an additional intricacy: some ecosystem features may be a result of residual selforganisational responses to prior environmental conditions that are no longer active.

We present a mathematical model in which an ecosystem of species, all at the same trophic level, compete for limiting resources according to Lotka-Volterra (LV) dynamics. In common with standard LV models, competition between species is modeled through interaction coefficients that summarise the extent to which species' niches overlap (Pianka, 1974). We extend this framework by allowing evolutionary pressure to affect species resource utilisation profiles, such that natural selection alters the pattern of species nicheoverlap, with the result that interaction coefficients change over evolutionary timescales. We alternately expose this system to two configurations of environmental forcing, each of which favours certain species over others (through variation in environmental carrying capacities, Figure 1). In the case where niche-space is saturated, decreases in competition with one species can only be achieved through increases in competition with another and species are under greatest selective pressure to minimise competition with those species with which they co-occur at the highest densities.

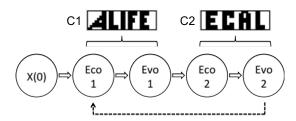


Figure 1: Schema of simulation showing key stages. An initial community of 232 species, X(0) is subjected to environmental conditions C1 (a pattern of differential carrying capacities). Ecological dynamics (Eco 1) are run until the system reaches an stable distribution of species densities (i.e. an attractor), at which point evolutionary effects are applied (Evo 1). The process is repeated for environmental conditions C2. As the distribution of carrying capacities in C1 and C2 is arbitrary, we have chosen patterns that correspond with two events that occur in alternate years.

Under these conditions, ecosystems that are exposed to multiple patterns of environmental forcing develop attractors for these configurations even when environmental forcing is lifted (Figure 2b). This property enables the system to recover these specific configurations, even from initial conditions that are ambiguous compositions of both of the historically experienced environments (Figure 2c). We find that the longer that an ecosystem remains under any configuration of forcing: i) the greater the disturbance it can withstand when forcing is removed and still return to the same composition of species; ii) the speed at which the system recovers from a disturbance increases; and iii) the longer the system can spend evolving at another attractor before the current attractor is 'forgotten'.

We recognise the presence of underlying organisational principles that enable these interesting system-level behaviours to emerge from local-level selection. We discuss how access to these principles has the potential to advance our understanding of ecosystem properties such as memory, robustness and resilience.

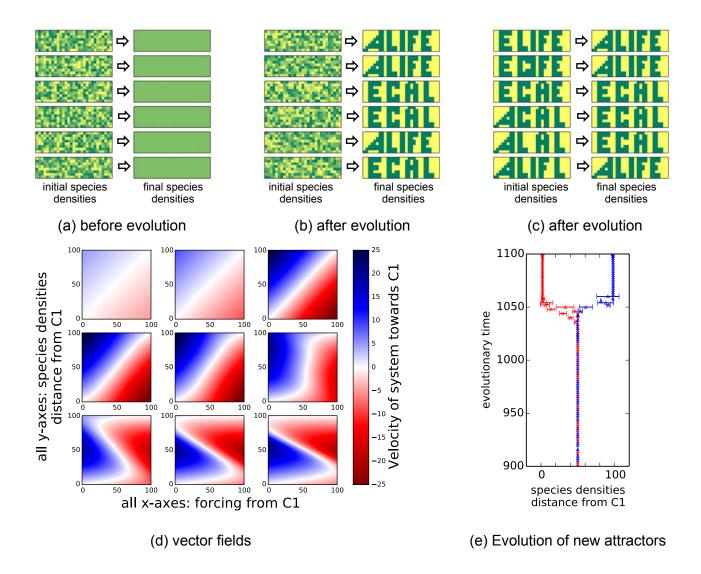


Figure 2: (a) Before the evolution of ecological interactions (and in the absence of any ecological forcing) the system has a single attractor, where all species are at the same density. After evolution of ecological interactions under alternating forcing, the system is examined again without ecological forcing. Now the system has two attractors corresponding to the exact configurations of environmental forcing it has experienced, meaning the system will return to one of these attractors from any set of random initial conditions (b) and even return to the nearest one of these attractors from initial conditions representing compositions of both the forcing patterns (c). These attractors are created as the system undergoes a fold bifurcation. The system's response to forcing is initially linear ((d): first row of panels). After an intermediate period of evolution ((d): middle row) the system's response to forcing is non-linear and the species densities develop increasing 'switch-like' convergence to the two attractors. At the final stage of system evolution ((d): last panel) ecological forcing is insufficient to move the system out its current attractor. (e) Monte Carlo analysis shows emergence of the two attractor state at around generation 1050.

References

Levin, S. A. (1998). Ecosystems and the biosphere as complex adaptive systems. *Ecosystems*, 1(5):431-436.

May, R. M. (2001). Stability and complexity in model ecosystems, volume 6. Princeton University Press.

Maynard Smith, J. and Szathmáry, E. (1997). The major transitions in evolution. Oxford University Press.

Pianka, E. R. (1974). Niche overlap and diffuse competition. Proceedings of the National Academy of Sciences, 71(5):2141–2145.

Evolving biological systems: Evolutionary Pressure to Inefficiency

Dominique Chu¹ and David Barnes¹

¹School of Computing, University of Kent, CT2 7NF, Canterbury dfc@kent.ac.uk

Abstract

The evolution of quantitative details (i.e. "parameter values") of biological systems is highly under-researched. We use evolutionary algorithms to co-evolve parameters for a generic but biologically plausible topological differential equation model of nutrient uptake. In our model, evolving cells compete for a finite pool of nutrient resources. From our investigations it emerges that the choice of values is very important for the properties of the biological system. Our analysis also shows that clonal populations that are not subject to competition from other species best grow at a very slow rate. However, if there is co-evolutionary pressure, that is, if a population of clones has to compete with other cells, then the fast growth is essential, so as not to leave resources to the competitor. We find that this strategy, while favoured evolutionarily, is inefficient from an energetic point of view, that is less growth is achieved per unit of input nutrient. We conclude, that competition can lead to an evolutionary pressure towards inefficiency.

Introduction

Much is now known about biological systems at the molecular level. There are countless databases that contain gigabytes of detailed information about biochemical networks, reactions, gene regulation, protein-protein interactions and much more. As far as biochemical reaction networks are concerned, most of the available information is about structural properties of these networks, i.e. which molecules react with which molecule, which protein represses/activates which gene and so on. At the same time, very little is known about the quantitative details of these reactions, i.e. how fast reactions proceed, how strong a gene is repressed or at what rate genes are expressed.

Recently, a large scale analysis of topological data has led to important insights into the design principles of living systems. The discovery of so-called network motifs(Alon, 2007; Kashtan and Alon, 2005; Mangan and Alon, 2003), i.e. over-represented local connectivity patterns of gene regulatory networks is but one example. These motifs were found to be not only statistically over-represented but also functionally significant(Alon, 2006). While much research effort has been expended to understand the significance of

these topological features, very little research has been done to understand quantitative details of biochemical reaction networks(Chu, 2013).

One of the conditions for being able to gain insight into the topological design principles of biological systems was the wealth of empirical available about them. Since there is not a comparable amount of information available about the parameter values of biochemical networks, it is only natural that much less is known about the quantitative design principles of natural systems. At the same time, it is likely that the values of parameters of biological systems contain very much biologically valuable information. They are a product of natural evolution and as such have to be assumed to reflect the adaptive pressures to which the system has been exposed and as such encode valuable biological information.

In order to understand the principles that guide the evolution of quantitative parameter values, it is not necessary to know the actual values of biological systems. Instead, a different approach based on synthetic evolution using evolutionary algorithms can be used. In this article we will take this approach. To do this we focus on a generic topological model of nutrient uptake, i.e. a model that only contains the structure of the biochemical reactions, but not their numerical parameters. We then use evolutionary algorithms to evolve parameters for specific conditions. Comparisons of a large number of runs will then enable us to draw some conclusions as to how parameters evolve. The hope is that these conclusions are valid beyond the specifics of the particular model we have chosen and provide insight about natural biological systems as well.

Our model does not describe any specific biological system, but it is a biologically plausible generic representation of nutrient uptake in bacteria and contains topological features that are widely used by bacteria. An important way for bacteria to take up nutrient is by importing nutrient molecules through specialised openings at the cell surface—so called *porins*. These porins are proteins and they tend to be specific to a particular nutrient type. So, a porin for one nutrient cannot be used to take up a different type of nutrient. In bacteria, these porins whose production requires energy

are only expressed by the cell when the relevant nutrient is actually present in the environment. A typical way for the cell to achieve this is to use the nutrient as an activator for the expression of the porin. Once imported into the cell the nutrient stimulates the expression of the gene coding for the porin (indeed, often it represses the repression of the gene, which amounts to stimulation). This very general scheme of porin activation is reflected in our model.

A typical feature of bacterial uptake system is that expression is demand driven and porins are only produced when they are needed. The evolutionary rationale for this is that gene expression requires resources that could be invested otherwise, for example to fuel growth. Moreover, there is finite space on the cell surface which limits the number of porins that can be expressed at any one time. It is also commonly observed that over- or under-expressing a gene often decreases the growth of the mutant strains. So, apparently, for many proteins there is an optimal rate of porin expression. At the same time, evolution has the ability to tune the rate of some biochemical reactions, including the rate of gene expression. It is therefore likely that the particular rates of gene expression and that of other bio-chemical reactions are fine-tuned by evolution.

This motivates the research question to be addressed in this contribution: How do the parameters of generic bacterial uptake systems depend on the adaptive pressures that led to their emergence. Moreover, given a set of adaptive pressures, is it possible to predict the parameters, or vice versa, given a set of parameters, is it possible to understand what adaptive pressures led to them? Finally, can the results obtained from the generic biologically plausible model provide any insight that is relevant for the real world.

To address these questions, we performed two types of artificial evolution experiments. Firstly, we evolved parameters (i.e. "solutions") on their own. We found that this results in uptake mechanisms that could turn most of the nutrient on offer into growth using a very low number of porins resulting in slow nutrient uptake and growth. We found this to be the most efficient mode of growth because it allows the cell to channel most nutrient into growth while minimising the amount of energy spent on the uptake mechanisms. In a further set of experiments we then evolved new solutions in competition with a previously evolved one.

The solutions obtained from these co-evolutions were different from the solutions evolved without competition. Rather than taking up nutrients slowly with a low number of porins, they evolved towards increasingly rapid uptake of nutrient (although not necessarily rapid growth). While this allowed them to grow fast it also means, as we will discuss below, that they grow inefficiently. Specifically, we found a clear trend that co-evolved solutions are less efficient than the original solutions that evolved without a competitor. However, within the chain of evolved solutions there was no clear further trend toward inefficiency. Hence, rather

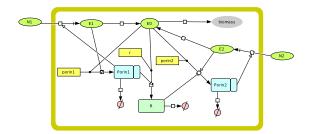


Figure 1: A schematic representation of the model.

than getting more efficient by competition, we found that coevolution leads to less efficient solutions, which is a counterintuitive at first. We will argue below that this pattern towards inefficiency is universal, in the sense that it does not depend on the specifics of the particular model, but would be true for a large class of nutrient uptake systems, including those of real organisms.

Furthermore, in our simulations we presented the simulated cells with two different types of nutrients of differing quality. We also added a structural motif into the model that would allow the cells to suppress take-up of the less valuable nutrient 2 in favour of the other. Indeed, we observed the evolution of the suppression of nutrient 2 uptake. However, surprisingly to us, the solutions did not use the motif offered, but instead came up with a different way of regulating the uptake of the less efficient nutrient.

The basic model

We present here a generic model of a bacterial uptake/metabolic system (see figure 1). The idea is that there are two sources of nutrients N_1 and N_2 . Uptake of these sources of nutrients requires specific porins, namely P_1 and P_2 respectively. Once taken up into the cell the nutrient becomes an internal source of energy $(E_1$ and $E_2)$ which can be converted into actual energy (or ATP), which we denote by E_0 . We assume that the uptake and conversion of nutrient follows Hill kinetics(Chu et al., 2011). The internal energy is converted either into porins (i.e. porin 1 and porin 2 abbreviated as P_1 , P_2) or into biomass (bm) which represents the results of bacterial growth.

We only determined the topology of this model which is designed such that the expression of porin 1 and 2 is activated by the presence of nutrient 1 and 2 within the cell (i.e. E_1 and E_2 respectively). The model topology does not by itself specify how strong this activation is. The strength of the activation depends on the parametrisation, which needs to be evolved. Indeed, there are many parameters that would effectively turn off the activation. The same is true for all other regulatory functions in the model.

An important feature of the model is that the expression of nutrient and the production of biomass require energy. Hence, the (*a priori* unspecified) parameter values for the expression rates of porins and the growth rate decide to what extent the resources (i.e. nutrient) is used to fuel growth and to what extent it is used to maintain the cellular uptake machine, i.e. how much is allocated to porin production.

It appears that there is an optimal allocation of resource to growth and the uptake mechanism. If the cell allocates no energy to uptake but all to growth, it will not be able to use any of the nutrients and hence it will not grow at all. On the other hand, if the cell allocates all of its nutrients into uptake, but none into growth, then it will be rich in nutrients, but never grow and hence never divide. In-between these two extremes there is one (or possibly several) optimal allocation. While it is clear from this argument that such an optimum exists, we do not know where it is and what it depends on.

Another important feature of the model is that the total number of porins in the system is limited. Porins in bacteria are located at the cell surface. They build openings there and selectively let molecules in and out of the cell. In real cells there is limited space on the surface to accommodate porins. This limitation is represented in our model by the term L (see below). It is a repressing term that reduces the expression of porins 1 and 2 as a (Hill-repressor) function of the sum of the concentration of both.

Finally, the model also features a repressor motif. The molecule R is expressed when there is porin 1 available in the cell and its sole purpose in the model is to repress the expression of porin 2. This sort of regulatory motif whereby a repressor is activated by some part of the system and represses another part of the system is commonly found in gene regulatory networks. The idea of introducing this motif is to enable the cell to evolve a repression mechanism for nutrient 2 when the (better) nutrient 1 is available.

The topology of the model can be summarised by these chemical equations:

$$N_{i} \rightarrow \epsilon_{P_{i}}E_{i}, k_{N_{i}}p_{i}\frac{N_{i}}{N_{i}+K_{N_{i}}}$$

$$E_{i} \rightarrow E_{0}, k_{E_{i}}E_{i}$$

$$P_{1}+E_{0} \rightarrow P_{1}, k_{P_{1}}\frac{E_{1}^{h_{P_{1}}}}{E_{1}^{h_{P_{1}}}+K_{P_{1}}^{h_{P_{1}}}}E_{0}L$$

$$P_{2}+E_{0} \rightarrow P_{2}, k_{P_{2}}\frac{E_{2}^{h_{P_{2}}}}{E_{2}^{h_{P_{2}}}+K_{P_{2}}^{h_{P_{2}}}}\frac{K_{R}^{h_{R}}}{R^{h_{R}}+K_{R}^{h_{R}}}E_{0}L$$

$$P_{0} \rightarrow k_{P_{0}}\frac{P_{i}}{P_{i}+K_{P_{i}}}$$

$$r+E_{0} \rightarrow R, k_{R}E_{0}\frac{E_{1}}{E_{1}+K_{R}}$$

$$\{P_{i}, E_{i}, R\} \rightarrow \emptyset, d_{\{P_{i}, E_{i}\}}$$

$$E_{0} \rightarrow bm, k_{C} \qquad (1)$$

where L is the space-limit which represents the fact that

there is limited space at the surface of cells to accommodate porins, given by

$$\frac{K_L}{(P_1 + P_2) + K_L}$$

The quality factor ϵ_{P_i} determines the quality of a nutrient and we set it to 1 for P_1 and 0.5 for P_2 . This means that one unit of nutrient 2 gives only 1/2 unit of biomass. Uptake and gene expression are assumed to follow Hill kinetics. While this is an approximation, in reality it has been found that Hill kinetics is a good description of the reactions described here. It is also widely used to model them and is a fairly simple approach. In all simulations reported here we keep the Hill exponent fixed at a value of 2, which is biologically plausible.

Evolving the system

In this article we evolve parameters for the topological model described by equation 1. Concretely this means that we evolve values for the kinetic parameters determining the system, including the Hill-constants (i.e. K_{N_i} and dynamic constants such as k_{P_i} . Note that we do not evolve the decay rate d which we keep fixed at 0.1, the Hill exponents (i.e. $h_x=2$), the relative value of ϵ_{P_i} (which we keep fixed at 1 and 0.5 respectively) and K_L which determines how much space there is for the porins in the cell. This latter parameter we set to 1. All other parameters are evolved and we allow them to take values between 0 and 15. In all simulations reported here the model is implemented as a system of differential equations. As a solver we use the general purpose numeric differential equation solver of the Maple computer algebra system version 16 for Linux.

The model was implemented as a co-evolutionary system, that is we have two different solutions compete for the same nutrient pool of N_i . This represents two different species of bacteria co-existing in the same environment. In practice this means that we used two sets of differential equations with two sets of the variables E_i , P_i , R, bm representing two different cell-types. Each set had their own kinetic parameters, yet their dynamics depended on one another via the shared nutrient pool. Of the two competing solutions, we ever only evolved one of those solutions, while keeping the other one fixed. Initially, we use as the fixed solution an "unfit standard solution" with all parameters set to 1. This solution supports no growth beyond the start-up allocation which is equivalent to 1 unit of biomass. Co-evolution is achieved by using previously evolved solutions as fixed solutions (i.e. "incumbents") in further evolutionary runs. In all simulations we set as the initial condition all variables to zero except for $P_i = 0.001, E_0 = 1$. This means that any solution can support a maximum of 1 unit of biomass even if it does not take up a single unit of nutrient.

During each evolutionary run only one of the solutions is evolved, while the other one is kept fixed at user-defined parameters. Co-evolution was achieved in a sequential manner, that is, one solution was evolved against a fixed solution. The evolved solution was then subsequently used as the fixed incumbent in a further evolutionary run. Co-evolutionary chains were obtained as follows;

- Evolve a first solution against an un-evolved base solution (all parameters set to 1).
- 2. Once the first solution is obtained, evolve a second solution against the first solution (which is kept fixed).
- 3. Create a third solution by evolving against the second solution (which is now also kept fixed).
- 4. Continue in this manner until no more solutions evolve.

To evolve the system we used a genetic algorithm with elitism. Individual solutions were represented as an array of real numbers in the range [0,15]. The population size was set to 50. The initial population consisted of random parameters within the range [0,15] sampled from a uniform distribution. As a fitness function we chose the biomass after 500 units of time. We found that 500 time units was large compared to the transient periods of the system, i.e. increasing this time did not change the results of the evolution.

As a selection algorithm we chose a fitness proportional selection. However, in every generation the best solution and a mutated version of it was allowed to proceed to the next population. The mutation and crossover rate was set to 0.8. Mutation was done by changing a random parameter by up to ten percent of its current value. If a mutation resulted in a value lower than 0.00001 or greater than 15 then the parameter was set to 0.00001 and 15 respectively. The amount of available nutrient was set to 10 for both nutrient types. The GA was implemented in Perl, but the fitness function was evaluated using Maple. Both the relevant Maple script and the Perl source code are available from the authors upon request.

We performed two different types of experiments. Firstly, we performed a simple evolution without competition (i.e. with the standard unit solution as competitor). Subsequently, we used the results of those evolutionary simulations to initiate a co-evolutionary chain, as described above. In practice we found that after a number of iterations no more fit solutions were found, in the sense that the total biomass produced for the evolving solution did not substantially exceed 1, i.e. evolution could not find solutions to outperform the incumbent. In this situation it was helpful to evolve a new solution by seeding the new evolutionary solution with the incumbent parameters, rather than starting from a random solution. However, even in this case, the co-evolutionary potential was limited.

Individual evolutionary runs were stopped either after 5000 generations or when a plateau of high fitness with no apparent further increases over time was reached, whatever

happened first. The presence of such plateaus was determined by visual inspection. In practice, it turned out to be a clear-cut case. A typical evolution would show rapid increases of the fitness at first, followed by fitness stagnation.

Results

Unconstrained evolution

We evolved a number of solutions without competitor. Figure 2 illustrates three typical results obtained from unconstrained evolution. It shows the amount of biomass over time obtained by simulating in Maple the best solution of the final population in the GA. It is part of the set-up that there is a limited amount of nutrient of 20 units divided across two types of nutrients. Since the second nutrient gives only half the growth of the first, at best the available resource can be converted into a biomass of 15 units under ideal conditions; the solutions also get a start-up energy equivalent to 1 biomass. Hence, in total the maximum they can reach is 16 biomass units.

It is apparent from figure 2 that most solutions evolved come close to the maximum attainable biomass, although there is some variation. Occasionally, we have also observed that solutions got stuck on a local minimum and did not discover the second nutrient source. This resulted in cells that would not take up any of the nutrient 2 and achieve only a level of about 10 units of biomass (data not shown). This indicates that the solutions were able to channel most nutrients into growth rather than using them for enzyme production. This high level of conversion was made possible by a very low assumed degradation rate of enzymes that allowed the solutions to grow at a slow rate.

The figure shows that the time required for achieving the maximal growth varies somewhat from solution to solution. The three example solution shown in figure 2 are representative for the range observed in all unconstrained evolutionary runs, Generally, we observed that these evolved solutions take up nutrient over a time period of 20 to 150 time units. There is a wide variation between the solutions that we obtained

Co-evolution

Co-evolution changes the nature of the solution obtained in very specific ways. The system as a whole offers a finite amount of resources and both solutions need to compete for the same two pots of nutrients. Hence, competition is not a zero sum game.

At the beginning of a co-evolutionary run the competitor will have random parameters and not be able to compete well against the incumbent. However, as new solutions are discovered the competitor evolves to outperform the incumbent. One way to do this is to consume the available nutrients faster than the incumbent. Ideally, the new solution has used up all of the nutrients before the incumbent can

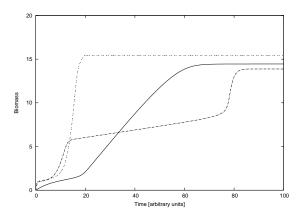


Figure 2: Three solutions of simple evolutions where the competitor is the maximally unfit solution.

do this preventing the latter from growing. Indeed, throughout all co-evolutionary runs we performed, this speed strategy emerged as one important way for solutions to undermine their competitors' abilities to grow. Co-evolution led to a sequence of increasingly fast solutions until a limit was reached and no more increases were possible. Note, however, that increased speed does not necessarily mean an increased growth-rate. Indeed, we observed a number of cases where growth was slower (i.e. occurred later) in the new competitor than in the incumbent but its nutrient uptake was still faster.

Figure 3 shows a typical co-evolutionary interaction. The first solution, which has been evolved against the unfit set of parameters, takes up nutrients slowly. This particular solution requires more than 100 time units to reach the final biomass. In contrast, the second solution is much faster and reaches its final biomass within 15 time units. Interestingly, the third solution, which is evolved against the second one, grows slower. Yet, a closer inspection shows that, while it produces biomass slower, its nutrient uptake is faster than that of the second solution. Hence, it leaves no nutrient to the second.

In all evolutionary experiments we performed we never found a case where a solution evolved to co-exist with its competitors, in the sense that *both* the incumbent and the competitor were able to take-up nutrient and grow. Instead, in all cases we considered, one solution came to dominate the other. However, there are cases where we observed the dominated solution to have some minimal growth, i.e. less than 1 biomass unit above the start-up energy.

Connected to this minimal growth of the dominated competitor we observed an interesting phenomenon. Figure 4 shows two simulations of a solution that we had obtained as a third solution during one of our co-evolutionary chains. Note that the graph does *not* show the evolution experiment, but an evaluation of the solution obtained from one of the

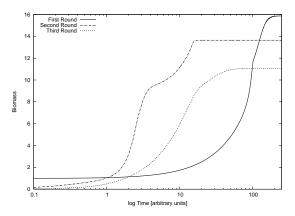


Figure 3: Three co-evolutionary solutions. The three curves show the growth of the solutions as a function of time. The curve labelled "First" is the original solution evolved against unfit parameters. The line "Second Round" was evolved in competition with the first and similarly the "Third Round" was evolved against the second. Time is shown in log scale to improve readability.

evolution experiments. The difference between the two runs in figure 4 is the competitor with which the evolved third solution competes. In one curve it is the standard unfit solution (which does not consume any nutrient) and in the other it is the second solution, i.e. the solution against which the third solution was evolved. In this particular case the second solution takes up a small amount of nutrient when competing against the third solution and grows roughly by 0.6 (data not shown) above its initial endowment. On the other hand the unfit solution, where all parameters are set to 1, does not take up any nutrient in competition with the third solution. Hence, one would assume that the growth of the third solution when competing with the second is lower than when competing with the standard solution. However, in reality, the third solution leads to a *higher* biomass in combination with the second solution than against the standard unfit solution. This is shown in figure 4. Increased nutrient uptake requires a higher level of investment into the metabolic machinery compared to the

Upon closer inspection this effect can be related to the usage of the second (less efficient) nutrient. In competition with the standard unfit solution the third solution does not use up all of the less efficient nutrient, but in competition with the second solution it does. This hints at the explanation for the observed effect. When competing with the standard unfit parameters the third solution has more nutrient 1 available. This additional nutrient leads to a higher production of porin 1 than when competing with the second solution. Note that there is a limit to the total number of porins for nutrient 1 and 2. Hence, if there is more porin for nutrient 1 produced then this means that less porin for

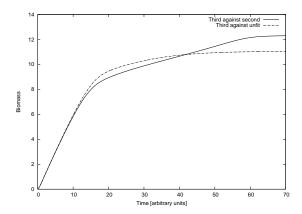


Figure 4: Comparing the third solution when evaluated against the unfit solution and against the second solution. Surprisingly, it does better against the (evolved) second solution than it does against the unfit standard solution.

nutrient 2 can be produced. Indeed, in the competition with the standard solution the porin for nutrient 2 is lower and tends to zero before all of the nutrient 2 can be taken up. In competition with solution two, on the other hand, the overall amount of porin 1 is lower which allows more porin 2 to be produced. The effect of this is that sufficient amounts of the porin can be produced to take up all of the available nutrient 2. Altogether, this leads to higher growth.

Based on this, one would expect that less biomass is produced by solution three in competition with the second solution than with the unfit parameters when the limit on the total number of porins is removed. To check this, we performed simulations where we removed the limitation (i.e. removed the factor L from equation 1). A comparison of solution three under these two different conditions then shows that indeed it develops more biomass when paired with the unfit solution than when with solution two (data not shown).

A comment on switching

In real bacteria there is a phenomenon called "diauxic growth." When bacteria are presented with two nutrient sources of different quality then they take up the good quality source first. Only when this one is exhausted will they take up the secondary source. From an adaptation point of view it is quite straightforward to make sense of this. Those cells that take up the good quality nutrient faster will be able to produce more offspring (because they have the better quality nutrient) and hence out-compete the others while at the same time leave less for their competitors. By the same reasoning, we expected to observe the emergence of diauxic growth in our artificial evolution experiments. Hence, we included a simplified mechanism to allow cells to suppress production of porin 2 when porin 1 is present in the cell. We specified that the regulator R has a suppressing effect on the

expression of porin 2, but requires porin 1 to be expressed itself. Given the right parameters, it should then be possible for diauxic growth to emerge.

In our simulations we found that the evolved solutions universally favoured porin 1 over porin 2, but they did not use a switching mechanism based on the regulator R. Instead the cells evolved other mechanisms to ensure that nutrient 1 is always taken up before nutrient 2.

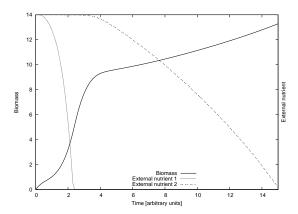


Figure 5: A second round solution and the amount of external nutrient available. Nutrient one is exhausted after around 2.5 time units. Only then the consumption of nutrient 2 starts. The control mechanism relies on limited space for porins. See main text for details.

We observed a small number of solutions that did not take up nutrient 2 at all. Amongst those solutions that did take up nutrient 2 a subset did not have any apparent regulation mechanisms, but simply took up nutrient 2 at a slow rate compared to nutrient 1, i.e. produced porin 2 at a low rate. This is only a mechanism in the most trivial sense. A more advanced, true mechanisms that frequently evolved was based on the limit on the total number of porins via the factor L in equation 1. The idea is as follows: If the porins for nutrient 1 are expressed at a higher rate than those for porin 2, then this leads to a higher rate of uptake of nutrient 1, further stimulating expression of porin 1. Since there is limited space, once a certain amount of porin is expressed, further expression of any type of porin is suppressed. Altogether, this allows porin 1 to increase its advantage and to crowd out porin 2 which is expressed at a low rate only. Yet, once nutrient 1 runs out, porin 1 is no longer produced and then porin 2 can be expressed.

While this mechanism effectively repressed porin 2, it limits by design the speed with which porin 2 can be expressed and hence it limits the uptake speed of nutrient 2. The ideal scenario for a bacterial cell would be to take up nutrient 1 rapidly, then switch and take up nutrient 2 rapidly. However, the simple regulatory mechanism via L relies on the production of porin 2 to be slower than that of porin 1

and therefore does not allow efficient repression of nutrient 2 uptake while nutrient 1 is still present *and* rapid uptake of nutrient 2.

This begs the question as to why the system does not accept the repressor R for the regulation of nutrient 2. The repression topology we used is a common gene regulatory motif in biology to control the expression of genes. Yet still, in none of the simulations that we performed it was used to regulate the expression of porin 2. We suspect that this simple regulatory motif is not effective in the regulation of porin expression. We conjecture that the underlying reason for the failure to evolve has to do with the difficulty of removing the repressor once nutrient 1 has run out. Further investigations are required to understand why this regulatory system is not effective.

The effects of competition

From the above analysis of the solutions it becomes clear that co-evolutionary pressure changes the nature of the solutions. The first solution, that is evolved against an unfit competitor tends to take up nutrient over a long time. Subsequent co-evolved solutions tend to take up nutrients, especially nutrient 1, over a much shorter time. The question is now why in the absence of competition solutions tend to evolve towards slow uptake. One possible explanation could be that there simply are more solutions (i.e. combinations of parameters) that take up nutrient slowly than there are solutions that take them up fast. Hence, in the absence of co-evolutionary pressure, evolution is more likely to discover slow solutions than fast ones.

Another interpretation, that does not necessarily preclude the first explanation, is that there is a functional significance to the slow speed with which nutrient is taken up. To understand whether this is the case, we considered the growth efficiency of solutions. To do this we defined a simple measure of efficiency given by the biomass divided by the total nutrient usage. According to this measure, a solution is more efficient if it requires less nutrient to grow to a given size.

In order to gain an insight into the nature of the solution we plotted the efficiency over time; see figure 6. A clear pattern emerged. The first solution that evolved against the unfit standard solution was always more efficient than subsequent solutions. For subsequent solutions, however, there is no clear trend towards further inefficiency. So, the fourth solution may or may not be less efficient than the third solution from the same co-evolutionary chain. Figure 6 shows the efficiency of three consecutively evolved solutions as an example.

There are again two ways to interpret this finding. One could assume that this trend towards inefficiency is merely an artefact of the particular modelling choices made, or that it is a more general phenomenon that is relevant for a large class of systems including real systems. We believe the latter is the case. Within our model, nutrient can only be converted

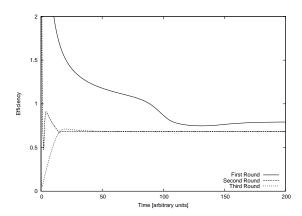


Figure 6: Comparing the efficiency of solutions during subsequent rounds. Efficiency is the amount of growth achieved for each unit of nutrient consumed.

into either biomass or into porins. The latter are necessary in order to take up nutrient. As long as there is no time-constraint on the system, it is sufficient for solutions to produce a small number of porins. It will take a long time to absorb all the available nutrient, but the investment into the metabolic machinery is low, so altogether the cell can grow efficiently. The major limiting factor here is the decay of nutrient which requires a certain production rate of porins to replace lost ones and keep the uptake stream constant. Up to that limit, slow growth is more efficient.

However, if a cell needs to compete with another one for resources, fast uptake is required, because otherwise the competitor takes up all the nutrient and nothing is left for the cell. Hence, competing cells need to take up nutrient rapidly. This is, however, inefficient. Uptake can only be achieved by a large number of porins concentrated into a small amount of time which entails a corresponding energy investment. Once the nutrient is used up, the porins no longer fulfil a function, and there is no return on their investment. Altogether, this results in an inefficient use of resources. Hence, fast nutrient uptake is inefficient independently of the specifics of the model assumptions, simply because it requires diversion of resources into porins.

Discussion and Conclusion

The current model makes a number of assumptions and simplifications. For example, the "infinite population" assumption implicit in the use of differential equations is of limited relevance for biological systems which are known to exhibit substantial noise at the molecular level. A deeper analysis of the system presented here would have to take into account stochastic fluctuations originating from the discrete nature of biochemistry. Yet, simulating such discrete systems is much more difficult than solving differential equations. Hence, for a first analysis differential equations provide a good trade-

off between feasibility and accuracy.

By using our model we found that taking up nutrients slowly is most efficient, but not necessarily the best strategy. Yet, in the absence of competitors the slowest possible growth is the most efficient one. In the hypothetical case of a continuous system with no protein breakdown an infinitely slow take-up rate corresponding to an infinitely slow expression rate of porin would be ideal. In more realistic models that include decay of components, there is an optimal rate of porin creation rate which depends on the rate of porin breakdown. The conclusion is that a group of clonal cells does best when growing very slowly, because then it expends the least amount of energy on maintaining the uptake machine. Only in competition with other cells will faster uptake rates be beneficial.

Not included in the above picture is the cost and the speed of computation. If we allow dynamically changing environments in the model then the picture changes. Environmental changes need to be sensed by the cell which then has to make internal adjustments based on the sensed changes. In the simplest case this is simply the presence and absence of nutrients. It can be shown that the speed with which these adjustment can be made depends directly on the breakdown rate and the speed of uptake. It has been shown recently(Chu et al., 2011) that slow uptake entails a limited ability to adjust to external conditions. On the other hand, faster uptake and growth is required to "compute" changes in the external environment effectively. Doing so comes at a cost in terms of additional nutrient that needs to be expended. Moreover, a hypothetical cell with no breakdown of components is not able to switch to a new state, simply because it is not able to forget its previous state. Say, at some point there are only porins of the first type in the system and these porins occupy all of the available surface, so that no more porins can be created. If then the nutrient of the first type is used up, the cell cannot express any other porins. As such it would miss out on growth opportunities. Similarly, if it can break down porins only slowly, then it will only be able to react slowly to changes in the environment. The conclusion from this is that extremely slow growth is only realistic for populations that live in constant environments that do not require any regulation.

So, in many ways the assumptions that we made in this contribution are somewhat unrealistic with respect to real biological system. Yet still, we think that the conclusions we reached are relevant. While ultra-slow speed will not be achievable in real systems, it is still likely the case that slower growing cells would be more efficient that faster growing ones simply because they will have lower rates of resource wastage. Yet, when in competition with other cells, then the slowest growth rate is no longer feasible and the cell has to invest a high amount of resource for growth. While the details of the evolutionary dynamics will be more complicated in real cells, and the particular trade-offs will be

more involved, the underlying fact that competition requires fast growth and that fast growth is inefficient is likely of very wide general applicability and relevant for our artificial cells and real biological cells alike.

Biological systems are commonly thought of as being optimal. The reasoning is that intense competition between cells will drive biosystems over time to fine-tune their internal processes to a point where resource usage and allocation is most "efficient." There are a number of well known problems of this optimality assumption. The best known one is that in evolving systems non-optimal, even slightly detrimental traits may piggy-back on advantageous traits and establish themselves in that way. Or, even in very simple fitness landscapes, constant mutational pressure will push the population away from any theoretical optimum generating a *quasi-species*(Eigen and Schuster, 1979). As a result of these and other similar effects biological systems cannot be assumed to be tuned perfectly to an optimum.

Our experiments show an additional biological driver towards inefficiency based on competitive co-evolution. Our results contradict the intuition that competition leads to efficiency. Under some circumstances biological systems are driven away from their most optimal mode of operation. One can now speculate whether or not the same effect applies in other competitive systems, such as in economics where it is routinely argued that competition to the most efficient allocation/use of resources. At least with respect to bacterial growth, our experiments seem to indicate that this is not necessarily so, but competition could lead to less efficient solutions rather than more efficient ones.

References

- Alon, U. (2006). An Introduction to Systems Biology: Design Principles of Biological Circuits. Chapman & Hall.
- Alon, U. (2007). Network motifs: theory and experimental approaches. *Nature Review Genetics*, 8(6):450–461.
- Chu, D. (2013). Evolving parameters for a noisy bio-systems. In 2013 IEEE Symposion series on Computational Intelligence.
- Chu, D., Zabet, N., and Hone, A. (2011). Optimal parameter settings for information processing in gene regulatory networks. *BioSystems*, 104:99–108.
- Eigen, M. and Schuster, P. (1979). The Hypercycle: A Principle of Natural Self-Organization. Springer-Verlag, Berlin, Heidel-Berg and New York.
- Kashtan, N. and Alon, U. (2005). Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Science USA*, 102(39):13773–13778.
- Mangan, S. and Alon, U. (2003). Structure and function of the feed-forward loop network motif. *Proceedings of the Na*tional Academy of Science USA, 100(21):11980–11985.
- New, A., Cerulus, B., Govers, S., Perez-Samper, G., Zhu, B., Boogmans, S., Xavier, J., and Verstrepen, K. (2014). Different levels of catabolite repression optimize growth in stable and variable environments. *PLoS Biol*, 12(1):e1001764.

Artificial Evolutionary Ecosystems

Evolution of animats following a moving target in an artificial ecosystem

Joachim Erdei¹ and Borys Wróbel^{2,3}

¹Department of Algorithms and System Modelling, Gdańsk University of Technology, Gdańsk, Poland
²Systems Modelling Laboratory, IO PAS, Sopot, Poland
³Evolving Systems Laboratory, Adam Mickiewicz University, Poznań, Poland
erdei@evosys.org, wrobel@evosys.org

Abstract

Many biological animals, even microscopically small, are able to track moving sources of food. In this paper, we investigate the emergence of such behavior in artificial animals (animats) in a 2-dimensional simulated liquid environment. These "predators" are controlled by evolving artificial gene regulatory networks encoded in linear genomes. The fate of the predators is determined only by their ability to gather food and reproduce—no subjective function is used to select the best individuals. Food is delivered to the environment by mobile animats who are not evolved ("prey"). Our results show promise for evolving more complex behavior relevant for nanorobotics, swarm robotics, and research on the evolution of simple cognitive abilities (minimal cognition).

Introduction

The ability to find resources (such as food) is common even among the simplest organisms. Even unicellular organisms, such as bacteria, are able to sense the concentration of various chemical substances (which can serve as food) in liquid environments, and move to a place where the concentration is higher. This phenomenon is called chemotaxis, and is an example of a minimally cognitive behavior (Beer, 1996).

Minimally cognitive behaviors attract considerable interest in Artificial Life, as a possible stepping stone towards more advanced cognitive skills (Wróbel, 2012). In many such experiments, artificial neuronal networks (ANNs) were used to control artificial organisms (animats; bio-inspired robots). In one such example of the state of the art approaches to evolution of simple behavior, virtual robots—resembling arthropods—searched for prey and avoided predators using vision (Palmer and Chou, 2012).

Yet, as we mentioned above, minimal cognition can be observed in biology even in unicellular organisms. One way in which single cells process information is with intracellular signaling networks (where nodes correspond to proteins, such as kinases, or small molecules, called transmitters; a state of the node—to concentration of, for example, a phosphorylated form of a protein, or concentration of a transmitter; and edges—to regulatory interactions). Such biological networks can be seen as computational devices, and

this goes also for gene regulatory networks. Here, nodes are genes and the edges are regulatory relationships (in which a gene product regulates the activity of a gene).

Artificial gene regulatory networks (aGRNs) are a model of computation inspired by such biological networks, and have been used in experiments on the evolution of directional movement in virtual mobile robots (Quick et al., 2003; Zahadat et al., 2010). The model of aGRNs we will use in this work, called GReaNs (Wróbel and Joachimczak, 2011) has been applied before for such tasks, with a genetic algorithm as a model of evolution of either unicellular animats (Joachimczak and Wróbel, 2010; Wróbel et al., 2012), or virtual soft robots (Joachimczak and Wróbel, 2012; Joachimczak et al., 2012).

Many multi-agent simulation platforms have been developed recently. BREVE (Klein, 2003) includes rigid body simulation with friction and collision detection. It proved to be useful for simulation of flocking behavior in 3D space (Spector et al., 2003). MASON (Luke et al., 2005) has been used in diverse applications including swarm multi-agent system in continuous and discrete space. DANCE (Shapiro et al., 2005) allows to model complex physical systems including body deformation. Nonetheless, neither of these platforms allow to model diffusion with efficient storing of chemical concentrations. Therefore, we developed a simple simulated environment, where artificial organisms (animats) compete for limited resources, reproducing only when they accumulate enough of them internally.

We used this platform in our recent work (Erdei et al., 2012, 2013) where we analyzed the evolution of chemotaxis. In this previously published work, we did not use a genetic algorithm—we believe that it is an imperfect model of a biological evolution, since it requires a subjective function to evaluate fitness, an emergent property in biology. Nonetheless, when we considered evolution of behavior of animats that could prey on other animats (Erdei et al., 2013), the results were somewhat disappointing. Although the prey produced a diffusing chemical, the predators failed to evolve the ability to use the gradient of this chemical to track the prey. Rather, the predators searched for immobile food

(from which another chemical diffused), and then waited for the prey attracted by food.

In the present work we analyze if tracking of mobile prey can evolve in our artificial life platform. As previously, the animats are controlled by evolving aGRNs (using GReaNs). The task considered here—tracking a movable resource—corresponds to following a leader in swarm robotics, or chasing prey by a predator in biology. Only the predators ("followers"), however, will be evolved—the movable resource will move randomly in the experiments we will describe further on, after providing a brief recapitulation of GReaNs in the next section.

Artificial environment with diffusible substances

The model of an artificial liquid environment we will describe is an extension of the model we presented previously (Erdei et al., 2012, 2013). The model allows to simulate diffusion of substances, and movement of artificial organisms (animats) in 2-dimensional environment. The environment has no boundaries (i.e., it is a two-dimensional torus)—an animat which crosses the left border appears on the right side. While organisms occupy real-valued positions, and concentration of a substance (which can serve as food or scent) is modeled as a function of real-valued position, the concentration is stored in the software in a space-discrete data structure (quadtree; Finkel and Bentley, 1974).

This structure divides space into squares of various sizes; each square corresponds to a tree node, with the square representing the whole space at the root. Each square can be divided into 4 smaller squares (when the concentration or its gradient is high), of equal size. Thus each node can have 4 children. The leaves store the information about concentration of a substance used for modeling diffusion of providing the actual concentration to the animats (at real-valued positions, with bilinear interpolation (Gribbon and Bailey, 2004); the concentrations for squares' vertices, and centers of vertices in some situations, are calculated as an average of adjacent squares' concentrations weighted by distances to squares' centers). To simplify calculation of diffusion, we imposed a rule that each square can only border on a square of the same size, a square 4 times larger in area, or a square 4 times smaller in area (see Fig. 1).

A quadtree is a compromise, in terms efficiency and precision, between dense and sparse grids, because during simulation the squares are divided (approximating a dense grid) where the concentration (or the concentration gradient) of a substance is high. Also, the squares are divided to the smallest possible (1 square length unit in size) where animats are and where substances are released. Elsewhere, the squares remain large. The concentration of each substance is modeled separately using an individual quadtree structure. Then the concentration in each point inside a square is calculated as a bilinear interpolation using these values calculated for

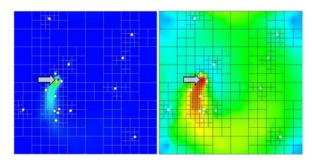


Figure 1: Modeling diffusion in 2-dimensional environment using a quadtree with continuous gradient of diffusible substances. The concentration of food is represented as hue, from blue (minimal concentration) to red (maximum concentration), scaled linearly in the left panel and logarithmically in the right one. Maximum concentration is around the prey (violet circles indicated by arrows in both panels). The predators (white circles) sense interpolated concentration values at their exact locations.

vertices.

Diffusion of substances between two adjacent squares follows the Fick's law:

$$\Delta P = \frac{c \cdot d}{D} \cdot (S_2 - S_1) \cdot \Delta t , \qquad (1)$$

where ΔP is the change in the amount of substance (by which the concentration will increase (decrease) in square 1 and decrease (increase) in square 2 if the current concentration is greater (lower) in square 2), c is the coefficient of diffusion (0.1 in the experiments described here), d is the length of the common edge, D is the distance between the centers of the squares, S_1 and S_2 are the current concentrations in both squares and Δt is the duration of the simulation step. Since each square stores concentration (not the total amount), the concentration in square X is changed by $\Delta S = \frac{\Delta P}{Ax}$ where A_x is the area of square X. At every time step, substances not only diffuse, but also degrade exponentially

$$S(t_0 + \Delta t) = S(t_0) \cdot g^{\Delta t} , \qquad (2)$$

where S(x) is the concentration of the substance in given square in time x, Δt is the duration of the simulation step, g is the degradation coefficient (in the experiments described here, 0.98 or 0.97), and t_0 is the previous moment of time.

Unicellular artificial organisms (animats) controlled by aGRNs

Animats are circular, and all are 1 length unit in diameter. Each is one cell with two flagella (Fig. 2), which work like engines (i.e., generating thrust). Flagella's activations are continuous (a value between 0 and 1) so an animat can move slower or faster depending on its strategy. When both flagella are fully activated, an animat accelerates by $\frac{1du}{(\Delta t)^2}$,

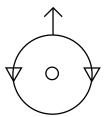


Figure 2: A unicellular animat with one sensor (circle) and two actuators ("flagella"; triangles). An arrow indicates the direction of movement when both actuators are activated equally.

where 1du is one distance unit, and Δt is the duration of the simulation step. When one flagellum is fully activated and the other is not activated at all, an animat accelerates its rotation by $\frac{1rad}{(\Delta t)^2}$. An animat experiences linear and angular friction which causes linear and angular drag proportional to its velocity squared. If animats collide, they bounce away from each other (in Erdei et al., 2012, 2013, collisions were not modeled).

There are two kinds of animats. One kind—the "prey"—have hand-written controllers which make them move in random trajectories. Prey do not eat or gather any substances, spend energy, die, nor reproduce. They deliver food to the environment (1 unit of food in each time step) that serves two purposes: it can be sensed by the second type of animats ("predators"), which also use it as a source of energy. In other words, prey are mobile food sources for the predators. The goal of our evolutionary experiments was to evolve predators which can follow the prey. We designed the experiments so that the only efficient way to obtain energy, necessary for reproduction, is to follow the prey. Using terminology taken from swarm robotics, the goal is to evolve animats that can "follow a leader".

In contrast to our previous work on controlling unicellular animats with aGRNs (Erdei et al., 2012, 2013), where animats had two sensors on two sides of the cells, predators in the present paper have each only one sensor for food. The sensor senses the local food concentration at the location which correspond to the cell center (Fig. 2).

A predator is controlled by an aGRN encoded in a linear genome. The model we use, called GReaNs, has been described elsewhere (e.g., Joachimczak and Wróbel 2008), and has been previously used to evolve controllers of animats (Joachimczak and Wróbel, 2010), cells in a multicellular body developing from a single cell (Joachimczak and Wróbel, 2008), and cells in a body that both develops and moves in an artificial environment (Joachimczak et al., 2012). Briefly, a genome is a list of genetic elements of 4 types: 2 types of *promoters* (*internal* and *output*), and 2 types of elements coding for *products* (*internal* and *input*). Each element is defined by 4 numbers, specifying its type, a sign, and two coordinates in two-dimensional affinity space.

A series (at least one) of internal promoters followed by a series (ditto) of product-coding elements is a *regulatory unit*. An output promoter is treated as a regulatory unit with a virtual single product (not encoded in the genome explicitly and which cannot bind to promoters), whose activation level translates here to the activity of one of the animat's actuators. There are 2 possible input products: one of them (i_1) is the value received by the food sensor (s) preprocessed by the logarithmic function:

$$i_1 = \frac{1}{7.5} \cdot \log_{10}(s) + 1$$
, (3)

and the second is always set to 1.

All products have continuous concentrations (between 0 and 1). An aGRN can be seen as a graph where vertices correspond to regulatory units, and edges correspond to product-promoter regulatory relationships. The product that regulates a regulatory unit does not need to belong to a different unit—self-regulatory relationships are permitted. When a product and a promoter have the same signs, the relationship contributes to the increase of the concentrations of all the products coded by the regulatory unit to which the promoter belongs. When the signs are different, the relationship contributes to the decrease of these concentrations. The strength of the relationship (the weight associated with the edge) depends on the Euclidean distance between the position of points (specified by the elements' coordinates) in an abstract 2-dimensional space. (This abstract space has nothing to do with the space in which the animats move, it is used solely to model chemical interactions between products and promoters.) When two points overlap, the relationship is the strongest possible. When they are further away than a prespecified distance threshold (5 units), there is no regulatory relationship. The concentrations of products coded by a regulatory unit (L_{Ω}) depend on their previous concentration and the concentrations of all the products with product-promoter relationships with the promoters belonging to the unit:

$$L_{\Omega} = \frac{2}{1 + e^{-(\sum_{j=1}^{J} \sum_{k=1}^{K} L_{k}(m_{k} \cdot m_{i} \cdot \frac{10 - 2d_{k,i}}{10d_{k,i} + 1}))}} - 1, \quad (4)$$

where J is the total number of promoters, K is a the total number of products and inputs in the genome, L_k is the concentration of the product k, $d_{k,i}$ is the Euclidean distance in the abstract space modeling product-promoter interactions, and m_k and m_i are element signs.

The animat gathers food $(0.75 \cdot S \cdot 1vu)$ food in each time unit, where S is food concentration at animat's location, and 1vu is 1 volume unit), and stores it internally. This food (energy) is spent on animat metabolism and movement:

$$\Delta M_t = M_b + M_m \cdot \frac{fa_1 + fa_2}{2} \,, \tag{5}$$

where M_t is the food spent in a time step, M_b is the base metabolism cost (0.003), M_m is the movement cost (0.004), fa_1 and fa_2 are flagella activation levels (values between 0 and 1).

If all food stored internally is spent, the animat dies and decays immediately, providing 4 units food to the environment. When stored food reaches 7 units, the cell divides, and two new animats are created. Division uses 4 units of food, and the rest of food stored by the parent is divided equally between the progeny. As in our previous paper (Erdei et al., 2013), an exact copy of the parent's genome is passed to one cell, and a mutated copy to another. The inheritance without mutation by one cell is inspired by the way elitism is achieved in microbial genetic algorithms (Harvey, 2011). This cell inherits the internal product concentrations from the parent. The cell that receives a mutated genome copy starts with internal product concentrations set to zero—this is necessary because the number of the products may change after the genetic operators are applied.

The mutational process consists of two parts. First, each individual element can change its type, with probability 0.01, or its sign, also with probability 0.01. With the same probability (0.01), both coordinates of an element are modified by a random value from a normal distribution ($\mu=0$, $\sigma^2=5$). Second, the element can originate (with probability 0.002) a duplication or deletion of a chunk of the genome. The number of elements copied (to a random location chosen from the uniform distribution) or deleted is sampled from the logarithmic distribution (with mean 10).

Evolution of tracking strategies

We conducted two sets of experiments, which differed in the speed of food degradation (degradation coefficients 0.98 and 0.97). The size of the environment was 256 length units across. Each experiment consisted of 7 phases. In each phase a group of predators was placed in the environment with specific number of prey (Table 1).

At the beginning of the first phase the environment was populated by 1000 predators with randomly generated genomes and 24 or 30 prey animats (Table 1). After 1 million (in the first phase) or 3 million (in all the subsequent phases) time steps, the number of prey was decreased. The prey count was high initially to enhance the survival of initially inefficient predators, and decreased afterwards so that the higher selection pressure prevents stagnation. We have conducted 20 independent evolutionary runs for each of two values of food degradation coefficient, saving the predator populations at the end of each phase (Fig. 3 and Fig. 4).

After the run ended, the saved populations were tested. In each test, only a relatively small number of predators was used because otherwise bouncing between efficient animats staying close to the prey might influence the results. To make the results as representative as possible for the actual populations, we first chose randomly 500 animats from each

Table 1: The number of prey in consecutive phases of the experiments for different food degradation coefficients.

Phase index	Number of prey for	
r mase mucx	coeff 0.98	coeff 0.97
1	24	30
2	16	24
3	9	20
4	7	16
5	5	12
6	3	8
7	1	4

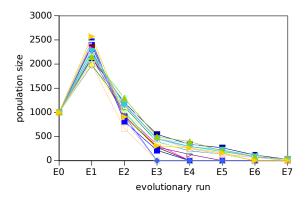


Figure 3: A size of the population for each evolutionary run after each phase (1—7; 0th phase is just 1000 animats which started the first phase) for degradation coefficient 0.98.

(with repetitions only if a population size was smaller than 500), and then separated this sample into 5 equal subsamples. These 100 predators were placed in the environment with 1 prey. During testing, the cells did not die or reproduce (and thus did not evolve). After 100 000 time steps, the mean distance at each time step between the predators and the prey was calculated, and the results for 5 subsamples were averaged. We also measured animats' progress in terms of the speed of gathering energy (the amount of eaten food decreased by metabolism costs) during the evolutionary runs. The initial population was used for comparison, but only one 100 000-step test with 100 predators and 1 prey was used for each run. In some runs no animats survived to the end of the last phase. For instance, only in 8 (for the coefficient 0.98) and 6 (for 0.97) runs there were surviving animats at the end of the penultimate phase (E6).

The results (Fig. 5, Fig. 6, Fig. 7, and Fig. 8) demonstrate that the largest evolutionary progress was made in the first 3 phases. After that the prey following behavior evolved slower or even stagnated. It is possible that the animats cannot follow the prey any closer, because there is a trade-off between moving closer and being pushed away by bouncing. Moreover, sensing changes in the direction the prey moves cannot be immediate (it requires diffusion), and we

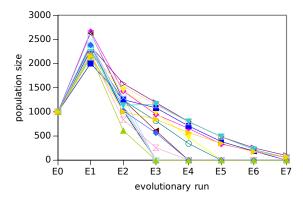


Figure 4: A size of the population for each evolutionary run after each phase (1—7; 0th phase is just 1000 animats which started the first phase) for degradation coefficient 0.97.

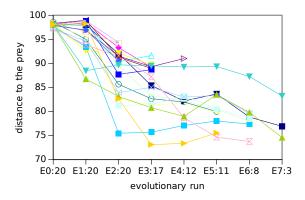


Figure 5: An average distance to the prey over 100 predators in the test environment with degradation coefficient set to 0.98. En for non-zero n is a population after n^{th} phase. E0 is a random population which starts the first phase. The number of populations tested (populations with survivors after each phase) is listed after a colon.

made the task even more difficult by giving the animats only one sensor. Even with those limited sensory abilities, the prey chasing behavior did evolve in the predators (Fig. 9).

Discussion and future work

We present an artificial life system where artificial organisms (animats) evolve without a genetic algorithm. The fitness is implicit and depends on the ability of an animat to find limited resources (food), necessary for reproduction with mutation. We demonstrate that in this system it is possible to evolve animats ("predators") that can follow a food source ("prey") that moves in a random trajectory, even though the predators had only one sensor of food diffusing from the prey.

The setting we used—one sensor, two actuators—is relevant for biological chemotaxis by small (not necessarily unicellular) organisms that cannot sense chemical gradients

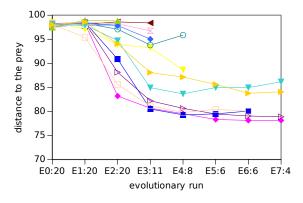


Figure 6: An average distance to the prey over 100 predators in the test environment with degradation coefficient set to 0.97. En for non-zero n is a population after n^{th} phase. E0 is a random population which starts the first phase. The number of populations tested (populations with survivors after each phase) is listed after a colon.

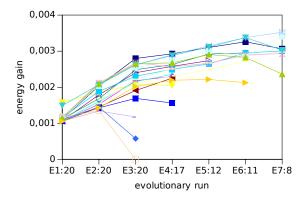


Figure 7: The average value of energy (the amount of eaten food decreased by metabolism costs) gained by one predator in one time step for each phase (E1—E7) of each evolutionary run (degradation coefficient set to 0.98). The number of populations is listed after a colon.

across their body, because their body is small relative to the differences in the concentration in their environment. Such organisms have to rely on temporal integration of information as they move. But the setting we used is not only biologically relevant—it is also applicable for cheap and small (possibly even nanoscale) mobile robots.

An even more challenging setting would be to use one sensor and one actuator; many biological organisms use actuators that can either propel the (possibly unicellular) body forward, or cause a random rotation of the body in space. This is a setting we plan to consider in the future. Even in the setting used here it will be interesting to analyze the fraction of the time in which the animat move predominately forward as opposed to local turning, in relation to the pattern in which the direction of the prey movement changes.

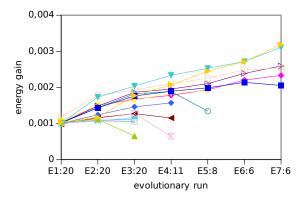


Figure 8: The average value of energy (the amount of eaten food decreased by metabolism costs) gained by one predator in one time step for each phase (E1—E7) of each evolutionary run (degradation coefficient set to 0.97). The number of populations is listed after a colon.

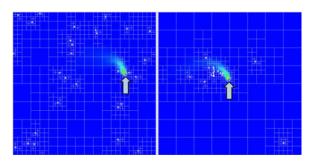


Figure 9: Prey tracking behaviour in evolved predators. The predators (white cicles) disperse all over the environment in the initial population (left), but in the runs with survivors in the last evolution phase (right), the predators remain close to the prey (violet circles indicated by arrows in both panels).

We hope that our system will allow us to test hypotheses relevant for biology and cognitive science, and evolve controllers useful in robotics. Therefore, other possible directions of future work include modifying position and number of sensors and actuators, changing how the sensors and actuators work, and introducing more challenging challenges or environments (e.g., seasonality or diversity of food sources, obstacles). This is the path towards evolution of complex cognitive abilities (Wróbel, 2012)—beyond minimal cognition displayed by predators evolved here.

Acknowledgements

This work was supported by the Polish National Science Center (BIOMERGE 2011/03/B/ST6/00399 to BW; JE acknowledges partial support of 2011/02/A/ST6/00201). We are grateful to Prof. Yaochu Jin for the discussions during BW's stay in his lab, supported by the European Commission FP7 Future and Emerging Technologies Proactive initiative: project number 270371 (COBRA). Computational

resources were provided by the Tri-City Academic Computer Centre (TASK), and the Interdisciplinary Centre for Molecular and Mathematical Modeling (ICM, University of Warsaw; project G33-8).

References

- Beer, R. (1996). Toward the evolution of dynamical neural networks for minimally cognitive behavior. In *From animals to animats 4: International Conference on Simulation of Adaptive Behavior*, pages 421–429.
- Erdei, J., Joachimczak, M., and Wróbel, B. (2012). Evolution of chemotaxis in single-cell artificial organisms. In *Proceedings* of *ICT Young 2012*, pages 185–190. Gdansk University of Technology.
- Erdei, J., Joachimczak, M., and Wróbel, B. (2013). Evolving gene regulatory networks controlling foraging strategies of prey and predators in an artificial ecosystem. In *Proceedings of the 12th European conference on Artificial Life, ECAL 2013*, pages 531–537. MIT Press.
- Finkel, R. A. and Bentley, J. L. (1974). Quad trees a data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9.
- Gribbon, K. T. and Bailey, D. G. (2004). A novel approach to realtime bilinear interpolation. In 2nd IEEE International Workshop on Electronic Design, Test and Applications (DELTA 2004), pages 126–134. IEEE Computer Society.
- Harvey, I. (2011). The microbial genetic algorithm. In *ECAL 2009: Proceedings of the 10th European Conference on Artificial Life*, volume 5778 of *Lecture Notes in Computer Science*, pages 126–133. Springer, Berlin / Heidelberg.
- Joachimczak, M., Kowaliw, T., Doursat, R., and Wróbel, B. (2012). Brainless bodies: Controlling the development and behavior of multicellular animats by gene regulation and diffusive signals. In Artificial Life XIII: Proceedings of the Thirteenth International Conference on the Simulation and Synthesis of Living Systems, pages 349–356, Cambridge, MA. MIT Press.
- Joachimczak, M. and Wróbel, B. (2008). Evo-devo in silico: a model of a gene network regulating multicellular development in 3D space with artificial physics. In Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems, pages 297–304. MIT Press, Cambridge, MA.
- Joachimczak, M. and Wróbel, B. (2010). Evolving gene regulatory networks for real time control of foraging behaviours. In Artificial Life XII: Proceedings of the Twelfth International Conference on the Simulation and Synthesis of Living Systems, pages 348–355, Cambridge, MA. MIT Press.
- Joachimczak, M. and Wróbel, B. (2012). Co-evolution of morphology and control of soft-bodied multicellular animats. In Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation, GECCO '12, pages 561–568, New York, NY, USA. ACM.
- Klein, J. (2003). Breve: A 3d environment for the simulation of decentralized systems and artificial life. In *Proceedings of* the Eighth International Conference on Artificial Life, ICAL 2003, pages 329–334, Cambridge, MA, USA. MIT Press.

- Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., and Balan, G. (2005). Mason: A multiagent simulation environment. *Simulation*, 81(7):517–527.
- Palmer, M. E. and Chou, A. K. (2012). An artificial visual cortex drives behavioral evolution in co-evolved predator and prey robots. In *Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference Companion*, pages 361–364, New York, NY, USA. ACM.
- Quick, T., Nehaniv, C., Dautenhahn, K., and Roberts, G. (2003). Evolving embodied genetic regulatory network-driven control systems. *Lecture Notes in Computer Science*, 2003:266–277.
- Shapiro, A., Faloutsos, P., and Ng-Thow-Hing, V. (2005). Dynamic animation and control environment. In *Proceedings of Graphics Interface 2005*, GI '05, pages 61–70, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada. Canadian Human-Computer Communications Society.
- Spector, L., Klein, J., Perry, C., and Feinstein, M. (2003). Emergence of collective behavior in evolving populations of flying agents. In *Proceedings of the 2003 International Conference on Genetic and Evolutionary Computation: PartI*, GECCO'03, pages 61–73, Berlin, Heidelberg. Springer-Verlag.
- Wróbel, B. (2012). Challenges for a-life approach to artificial cognition: in search for hierarchy of cognitive systems. In Artificial Life XIII: Proceedings of the Thirteenth International Conference on the Simulation and Synthesis of Living Systems, pages 599–600, Cambridge, MA. MIT Press.
- Wróbel, B. and Joachimczak, M. (2011). Using GReaNs (genetic regulatory evolving artificial networks) for 3-dimensional asymmetrical pattern formation and morphogenesis. In *Proceedings of DevLeaNN: A Workshop on Development and Learning in Artificial Neural Networks*, pages 26–28, Paris, France.
- Wróbel, B., Joachimczak, M., Montebelli, A., and Lowe, R. (2012). The search for beauty: Evolution of minimal cognition in an animat controlled by a gene regulatory network and powered by a metabolic system. In From Animals to Animats 12: The 12th International Conference on the Simulation of Adaptive Behavior (SAB 2012), volume 7426 of Lecture Notes in Computer Science, pages 198–208, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Zahadat, P., Christensen, D. J., Schultz, U. P., Katebi, S., and Stoy, K. (2010). Fractal gene regulatory networks for robust locomotion control of modular robots. In *Proceedings of the 11th International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, SAB'10, pages 544–554, Berlin, Heidelberg. Springer-Verlag.

Population and Evolutionary Dynamics based on Predator-Prey Relationship in 3D Physical Simulation

Takashi Ito¹, Marcin L. Pilat¹, Reiji Suzuki¹ and Takaya Arita¹

¹Graduate School of Information Science, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan takashi@alife.cs.is.nagoya-u.ac.jp

Abstract

Recent studies have reported that population dynamics and evolutionary dynamics, occurring at the different time scales, can be affected by each other. Our purpose is to explore the interaction between population and evolutionary dynamics using an Artificial Life approach based on a 3D physically simulated environment in the context of the predator-prey and morphology-behavior coevolution. The morphologies and behaviors of virtual prey creatures are evolved using a genetic algorithm based on the predation interactions between predators and prey. Both population sizes are also changed depending on the fitness. We see in the evolutionary simulations two types of cyclic behaviors corresponding to the short-term and long-term dynamics. The former can be interpreted as a simple population dynamics of Lotka-Volterra type models. It is shown that the latter cycle is based on the interaction between the changes in the prey strategy against predators and the long-term change in both population sizes, resulting partly from a tradeoff between their defensive success and the cost of defense.

Introduction

Evolutionary and ecological dynamics have usually been thought to influence each other asymmetrically. Evolutionary changes are usually a consequence of the environment while they occur over timescales that are too long to affect the dynamics of population size in the short term (Schoener, 2011). Therefore, most ecological models ignore evolutionary changes in the conspecific or other species, assuming a separation of timescales between population dynamics and evolutionary dynamics (Hastings, 1997).

However, this assumption is challenged by recent studies on "rapid evolution" in nature occurring on time scales comparable to those typical of population dynamics. In view of such studies, the concept of evolutionary time defined by researchers is changing rapidly (Schoener, 2011). Among them, Hairston et al. go the farthest in defining rapid evolution as a genetic change rapid enough to have a measurable impact on simultaneous ecological change (Hairston et al., 2005). With progress of related studies, many researchers have come to conclude that when rapid evolution occurs during the course of an ecological process, it can significantly change ecological predictions (Fussmann et al., 2007).

Accepting not only that ecology affects evolution but also that evolution affects ecology means that the transformed ecology might affect evolution, and so on, back and forth in a feedback loop, referred to as "eco-evolutionary feedbacks" (Post and Palkovacs, 2009) or "ecogenetic feedback" (Kokko and Lopez-Sepulcre, 2007). The feedback is not entirely straightforward and there are several challenging questions. Among them, the most fundamental one is about the importance of the feedback. It is claimed that only an extensive research effort involving multiple experimental approaches with long-term field experiments over a variety of ecological communities will provide the answer, while the investigations to reveal the role of such feedback are just beginning (Schoener, 2011).

We believe that the Artificial Life approach based on 3D physically simulated environments will provide valuable insights into the relationship and the interaction between population and evolutionary dynamics. The virtual creature models, following the pioneering study (Sims, 1994b), allow us to analyze the morphology-behavior coevolution in 3D environments in the individual level (Taylor and Massey, 2000; Miconi and Channon, 2006; Chaumont et al., 2007; Pilat and Jacob, 2008; Azarbadegan et al., 2011).

Population dynamics can be straightforwardly introduced into these types of models. Population dynamics depends on the fitness of individuals if the fitness represents the offspring number of them in the models, although almost all previous models of virtual creatures have a fixed number of individuals. Given enough computational power, we can observe and analyze the interaction between population dynamics and evolutionary dynamics in 3D virtual creature environments. What we want to emphasize here is that virtually embodied creatures can evolve unexpected traits of morphology and behavior as a result of the interactions with conspecifics or other species in a physically simulated world. Since the genes and parameters do not have explicit pre-defined functions, like they do in previous studies based on mathematical models (Yoshida et al., 2007), functional traits emerge naturally. This makes for more natural evolution and allows us to discuss their emergence in context of

Eco-Evolutionary Dynamics.

Evolutionary process can affect ecological dynamics at intra- and interspecies levels. This study focuses on the predator-prey interactions as the key element of ecological systems (Legreneur et al., 2012). Predation pressures in food chains shape diversity and functions of organisms (Agrawal, 2001). Predators employ various strategies capturing their prey, and at the same time, prey employ various protective mechanisms against their predators in nature (Edmunds, 1974), that can be regarded as the results of the coevolution between predators and prey. Some previous studies using the Artificial Life approach explored competitive coevolution (Sims, 1994a; Miconi, 2008). Our previous model focused on the morphology-behavior coevolution under the environments with a predator and a prey (Ito et al., 2012, 2013a). We extend it to explore the interaction between population and evolutionary dynamics in the context of the predator-prey and morphology-behavior coevolution.

In this paper, we evolve the virtual creatures of the predators and prey with the change in their population size in a 3D physically simulated environment. We analyze the relationship between the population size and the trait evolution and show their two different behaviors depending on the time scale. We then discuss and estimate these interaction and relationship in terms of the population and evolutionary dynamics.

Model

We use Morphid Academy, which is an open-source simulation system (Pilat and Jacob, 2008) to evolve virtual creatures in a 3D physically simulated environment. This virtual creature model is a simplification of Sims' Blockies model (Sims, 1994b) and is fully described in (Pilat and Jacob, 2008). The simplification in body and neural structure reduces the evolutionary search space and has been demonstrated to perform well for various evolutionary tasks. Morphid Academy has been previously used to successfully evolve virtual creatures for locomotion (Pilat and Jacob, 2008), light-following (Pilat and Jacob, 2010), and sustained resource foraging (Pilat et al., 2012). In addition, it has been used to evolve the various successful strategies in one-to-one interaction between a predator and a prey (Ito et al., 2012, 2013a). We performed double coevolution of morphologybehavior and predator-prey couplings, presented the emergence of various morphological and behavioral strategies of prey against predation by predators, and analyzed the dynamics of this coevolution caused by the two levels asymmetries (Ito et al., 2013b).

In this paper, to represent the interaction between the group of predators and prey, we simulate and evaluate every predator and prey individuals of the both population pools in a shared environment (Fig. 1). Each species evolves their traits and change their population size depending on their fitness.



Figure 1: The virtual creatures in the 3D physically simulated environment. The big creatures correspond to the evolved predators and the small creatures correspond to the evolved prey. The semitransparent creature represents a prey caught.

Agent

The agents are virtual creatures comprised of several 3D rectangular solid body parts connected with simple hinge joints. Their physical phenotype is developed from a directed graph (Fig. 2). The nodes represent body parts and the links represent joints. The genotype graph undergoes evolution through a genetic algorithm. We termed the root body part as the torso, and all the other parts as limbs. The controller of a virtual creature is a recurrent neural network embedded in body parts. There are three types of neurons: input, calculation and output. The input neurons represent sensory information from the environment, the computational neurons process the input and the results are fed into other computational or output neurons, and the output neurons as joint effectors power the joints, making the creature move.

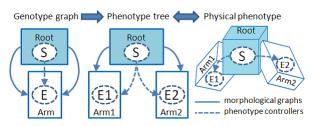


Figure 2: The development from genotype to phenotype.

The sensor of a creature detects a living creature belonging to the other species nearest to the virtual creature within a sensing range s (described in detail in (Pilat and Jacob, 2008) and (Pilat and Jacob, 2010)). Two measures are calculated by the sensor: the angle to the sensed creature with respect to the main orientation axis of the creature and the

distance to the sensed creature. These are combined into one value incorporating the sign of the angle and the value of the distance. The result is fed into the sensory neurons in the network. It is important for this experiment to use a small sensing range s. If the creatures can detect others in distant location with a large s, there is little effect of the density of prey on the predators because the predators are always possible to find any prey in the environment. It causes the decrease in the effect of the population dynamics on the evolution of traits.

Evaluation

In each generation, the predator group and the prey group are randomly positioned within a radius of C from the origin of the simulation space. Every agent is positioned above the simulation plane and allowed to free-fall due to gravity during a stabilization phase. Once they become stable resting on the ground surface, the an encounter phase for the evaluation begins and lasts for S simulation time steps.

Capturing is defined as the predator touching the torso of the prey with any of the predator's body parts. This definition represents that animals have a weak point in their body. The captured creature is disabled and cannot be sensed until the end of the simulation.

The evaluation value of each agent after the predator-prey encounter (the encounter phase) is calculated using Eq. 1 and 2. The evaluation value of a predator is defined by Eq. 1:

$$EV_{predator} = \begin{cases} \alpha_1 \times \{S_p + (1 - \frac{d_n}{d_0})\} & (\frac{d_n}{d_0} < 1) \\ \alpha_1 \times S_p & (\frac{d_n}{d_0} \ge 1), \end{cases}$$
(1)

where α_1 represents the parameter that adjusts the size of evaluation. S_p represents the number of successful predation by the focal predator. d_n represents the distance from the prey detected by the focal predator in the final simulation step and d_0 represents the distance between them when the focal predator began to detect that prey. This equation means that the predator that captures more prey and that tends to approach prey can obtain the larger evaluation value.

The evaluation value of a prey is defined by Eq. 2:

$$EV_{prey} = \begin{cases} \alpha_2 \times (1 - \frac{v}{\beta}) & (\text{escaped}, v < \beta) \\ 0 & (\text{caught}) \\ 0 & (\beta \le v), \end{cases}$$
 (2)

where v represents the volume, the parameter α_2 represents the parameter that adjusts the size of evaluation and β represents the coefficient for the maintenance cost of the larger volume. This definition of the evaluation value function means that the prey that has successfully escaped predation until the end of the simulation obtains a evaluation value but its size depends on its volume, which represents the cost for the maintenance of the large body. When the prey is captured or when the volume of the prey is larger than β , the evaluation value is 0.

Evolution and Population Dynamics

Two populations representing the predators and the prey are concurrently evolved for g generations using a genetic algorithm. Both population sizes P1 and P2 are changed simultaneously with the reproduction of the next generation. We used the following process for the genetic algorithm.

Each individual has an opportunity to reproduce some children by mating with another individual selected randomly. The expected value of the number of offspring n ($0 \le n \le M$) for the mating event is defined by the fitness based on the evaluation values of the both parents using Eq. 3:

$$Fitness = \frac{\text{Sum of the parents' evaluation value}}{D}, \qquad (3)$$

where the parameter D represents the difficulties of the reproduction. Note that if the number of the children exceeds the lower (upper) limit of the population $P1_{min}$ and $P2_{min}$ ($P1_{max}$ and $P2_{max}$), one (no) child is created by the parents, and thus the population is kept at the lower (upper) limit.

The parents produce n children through one of the genetic operators: copying (with the probability R_p), crossover (with R_c), and grafting (with R_g). A single point crossover exchanges parts of the genotype tree at the node level. The grafting operator grafts a randomly chosen subgraph from one parent onto another. A mutation is applied to the resulting child individual with the probability R_m , which applies small changes to the whole genome (with the probability 0.05 per gene). This change includes changes in the morphological node or link parameters, addition of morphological nodes, and the addition or removal of morphological links. The resulting creature is processed to remove unreachable nodes. The children of every individuals replaces the population. As a result, the population size is changed according the fitness of creatures.

Results

Experimental Setting

In the experiments as a first step in the new direction, we assumed that the prey population evolved as described above while the predator population did not evolve in order to understand the basic dynamics caused by evolution of one species. For this purpose, we pre-evolved the predators in preliminary experiments with random prey, and some successfully evolved predators were used to seed the initial population of predators. On the other hand, the random prey were used to seed the initial population. We also used the evolutionary parameters as follows: $R_p=1.0$, $R_c=0.0$, $R_g=0.0$ and $R_m=0.0$ for the predator population without evolution; and $R_p=0.8$, $R_c=0.1$, $R_g=0.1$ and $R_m=0.1$ for the prey population with evolution. We further used the following settings of parameters: g=1000, S=100000, S=300, C=1500, $P1_{max}=50$,

 $P2_{max} = 50$, $P1_{min} = 5$, $P2_{min} = 5$, $P1_0 = 25$, $P2_0 = 25$, $\alpha_1 = 10000$, $\alpha_2 = 10000$, $\beta = 50$, D = 6000 for predator, D = 6000 for prey, and M = 3.

Basic Behavior

We performed 10 trials of evolutionary experiments using those settings. We observed a similar tendency of the population dynamics in all the trials. Fig. 3 and Fig. 4 show the typical population dynamics and the change in the average fitness of predators and prey, respectively.

In these figures, the red line and the blue line represent the population size (fitness) of the predators and prey, respectively. In early generations, both populations were very low. At some point near the 250th generation, both populations increased suddenly and then started to fluctuate significantly. The change in the prey population was larger than that in the predator population in this period. After that, around the 600th generation, the both populations became very low, which is the similar to those in the early generations. Finally, around the 750th generation, both populations increased again. As this shows, it has the different two patterns of the population dynamics occurred alternately. In addition, we see that the fitness of both populations also changed in synchronization with their population size.

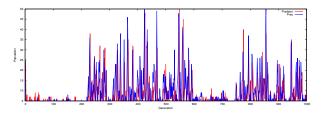


Figure 3: The typical result of the change in the population of the predators (red line) and prey (blue line).

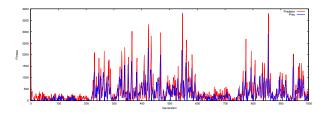


Figure 4: The typical result of the fitness of the predators (red line) and prey (blue line).

Population and Evolutionary Dynamics

We analyzed the relationship between the population and evolutionary dynamics of both populations. As the quantitative index of the evolutionary dynamics, we used the average volume of the body and tracked its evolution.

Short-Term Dynamics First, we focused on the period from the 300th to 400th generations that showed the typical population change in a short-term dynamics. Fig. 5 shows the populations of the predators (red line), prey (blue line) and the average volume of the prey (green line). We observed a periodic increase and decrease of the both populations and also observed that the change in the prey population preceded the change in the predator population. We estimated by time delay estimation (TDE) method (Chen et al., 2003) that the change in the prey population was followed with a time lag of about 2 generations by the change in the predator population.

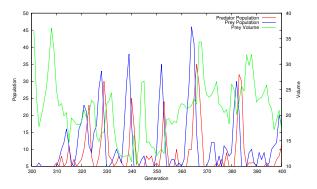


Figure 5: Population of the predators (red line) and prey (blue line) and the average volume of the prey (green line) from the 300th to 400th generations.

Long-Term Dynamics Second, we focused on the relationship in a long-term dynamics using a 30-period simple moving average of the indices in order to reduce the short-term fluctuations. Unlike those in a short-term period, the populations of the predators and prey changed simultaneously as shown Fig. 6. Both species had large populations at the 0th to 200th and the 600th to 750th generations and had small populations at the 200th to 600th and the 750th to 1000th generations. On the other hand, when the population of the prey gradually increased (decreased), the volume of the prey slightly decreased (increased) simultaneously.

Trajectory Analysis We can see the difference between these dynamics more clearly by focusing on the typical trajectories of population and evolutionary changes in a short-term dynamics (A and B) and a long-term dynamics (C and D), shown in Fig. 7. The trajectory of the predator and prey populations shows a typical cyclic behavior in the short-term dynamics (A), which is often observed in Lotka-Volterra systems (Vorterra, 1926; Lotka, 1932). In this case, the trajectory of the volume and the population of the prey showed no clear tendency (B), and the correlation coefficient was -0.30. This means that there are no interactions between the population and evolutionary changes of the prey.

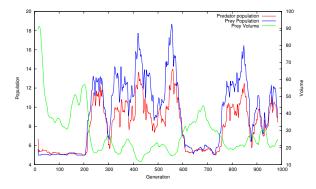


Figure 6: Population of the predators (red line) and prey (blue line) and the average volume of the prey (green line) smoothed by a 30-period simple moving average.

On the other hand, the graph C shows that there is a positive correlation between the predator and prey populations in the long-term dynamics. This is different from the one in A although we can still see small cycles in the trajectory. It also should be noted that, in the graph D, there is a strong negative correlation between the population and volume of the prey, and the correlation coefficient was -0.75. This means that there existed clear interactions between the population and evolutionary changes in this long-term dynamics. We also see the oscillations of these indices occurred repeatedly, keeping the correlation negative.

Thus, we can say that the mutual interactions between the population and evolutionary changes were observed only in the long-term dynamics.

This result suggests that the population and evolution dynamics had the different relationship between the short-term and the long-term dynamics. Especially, it is expected that the evolutionary dynamics strongly affected the population dynamics in a long-term dynamics, in contrast, it affected little in a short-term dynamics.

Influence of Population Dynamics on Evolutionary Dynamics

Next, we analyzed how the population dynamics affected the evolutionary dynamics of the prey. There are two factors as selection pressures on the prey: defense against predation and reduction in cost of defense, as is defined in Eq. 2. In our previous study (Ito et al., 2013a), we observed that the large volume was necessary for the prey to obtain the successful defensive strategies. Therefore, there is a trade-off between these two factors because the large volume is costly in our experiments. Theoretically, on some conditions, the change in the average value of a trait depends on the covariance between the trait and its fitness or, equivalently, the regression coefficient of fitness on the trait multiplied by the variance of the trait (Steven, 1998). In this model, we focus only on the

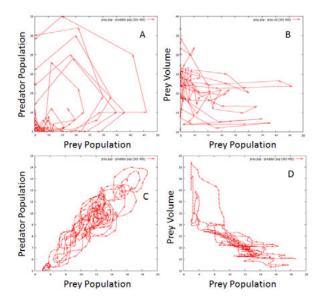


Figure 7: Typical trajectories of population and evolutionary dynamics. We used the data in Fig. 5 and Fig. 6 for plotting the graphs "A and B" and "C and D", respectively. (A) The predators and prey populations in short-term dynamics. (B) The volume and population of the prey in short-term dynamics. (C) The predators and prey populations in long-term dynamics. (D) The volume and population of the prey in long-term dynamics.

variance of the trait for simplification. We estimated from which selection pressure the prey population was affected by observing the relative variance RV_c , which is defined by following equation:

$$RV_c = \frac{V_c}{V_c + V_p} \tag{4}$$

where V_c is the variance of cost (= volume) and V_p is the variance of the successful escape from the predation.

Fig. 8 shows the predator population (red line) and the relative variance of the cost (black line). The relative variance of the cost increased while the size of the predator population decreased and the relative variance of the cost decreased while the size of the predator population increased. Therefore, the pressure of the predation tends to dominate in a large predator population and the pressure of the cost tends to dominate in a small predator population. It is adaptive for the prey to reduce the cost in an environment in which the predator population is small. The prey without defensive strategies with the lower cost can obtain the high fitness, because the probability of predation is low. In contrast, it is adaptive for the prey to increase the cost and have defensive strategies in an environment in which the predator population is large. If the prey escape from the predation by paying

the high cost, it is expected that they obtain the higher fitness than that in the case of paying a low cost because of the high risk of predation.

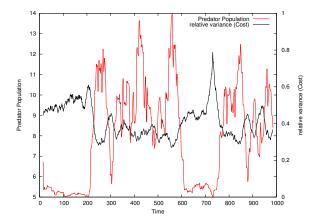


Figure 8: The predator population (red line) and the relative variance of the cost (black line).

Discussion

We discuss the two different types of the interactions between the population and evolutionary dynamics with different time scales observed in the previous experiments, which were illustrated in Fig. 9 and 10. In each figure, the outer and inner circular arrows represent the dynamics in the population level (i.e., change in the size of both populations) and the individual level (i.e., the evolution of the volume of the prey), respectively. The middle circular arrows represent the change in the target of selection.

Short-Term Dynamics

Fig. 9 shows their interactions in a short-term dynamics, which can be summarized as follows:

- 1. When the predator population is small, the prey population is increased by the low probability of predation.
- 2. The increase in the density of the prey population causes the increase in the probability of successful predation and the increase the in predator population.
- 3. The prey population is decreased by the large predator population due to the successful predation of the prey by the predators.
- 4. Finally, the predator population decreased caused the low density of the prey population. The both populations return to the step 1.

This cyclic dynamics corresponds to the one observed in the Lotka-Volterra systems, which is caused by the change in the

population density only. In this short-term dynamics, there seems no clear selection pressure for the trait of the prey. It is assumed to be due to the fact that population dynamics of both species were too fast for the prey population to adapt to, although it is too simplistic to conclude that evolutinary dynamics does not affect short-term population dynamics in this model.

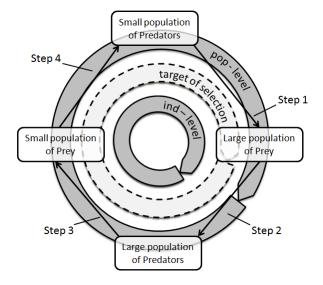


Figure 9: The interactions between the population and evolutionary changes in the short-term dynamics.

Long-Term Dynamics

Fig. 10 shows their interactions in a long-term dynamics, which can be summarized as follows:

- 1. When the predator and prey population are small, the probability of predation is low. Therefore, the reduction in the cost of defense becomes the target of selection.
- 2. The volume of the prey decreases.
- 3. Because the prey with the low cost body obtain high fitness and reproduce many offspring, the population of the prey increases. At the same time, the increase in the density of the prey population causes the increase in the predator population.
- 4. When the predator and prey population are large, the probability of the predation is high. Therefore, defense against predation becomes the target of selection.
- 5. The effective defensive strategies with the large volume of their body invade into the population of the prey.
- 6. Because of the high cost for their large volume as well as the high predation pressure, they get low fitness. Thus, the

population of the prey decreases, which further decreases the population of predator. Both predator and prey population return to the step 1.

In this long-term dynamics, there is enough time for the evolution process of the prey to adapt to their environmental condition because the change in the population of predators is relatively slow. Thus, the trait evolution of the prey occurred in response to the population dynamics of the predators, which further brought about the change in the population dynamics. This implies that there exists an appropriate timescale for the complex interactions between the population and evolutionary dynamics to emerge.

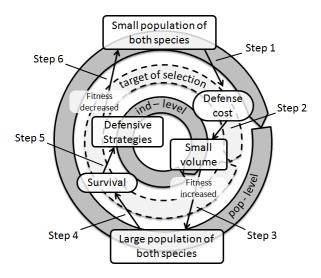


Figure 10: The interactions between the population and evolutionary changes in the long-term dynamics.

Recently, there have been various reports on the interactions between the population and evolutionary dynamics. As for the interactions in the predator-prey relationship, Yoshida et al. showed that there is a trade-off between the competitive ability and the defense against predation of the prey in rotifer-alga and phage-bacteria chemostats. The most competitive non-predator-resistant bacteria dominated initially, but as rotifer densities increased, the more predatorresistant bacteria dominated (Yoshida et al., 2004). They also showed that the predator or pathogen can exhibit largeamplitude cycles while the abundance of the prey or host remains essentially constant (Yoshida et al., 2007). They found that, in such a situation, there exist the cryptic cycles of interactions between these species through the rapid evolution of the frequencies of defended and undefended prey. Sanchez and Gore also demonstrated the presence of a strong feedback loop between population dynamics and the evolutionary dynamics of a social microbial gene, SUC2, in laboratory yeast populations whose cooperative growth

is mediated by the SUC2 gene (Sanchez and Gore, 2013). They showed that the eco-evolutionary trajectories of the population density and the gene frequency spiral in the density/frequency phase space. The long-term dynamics observed in our experiments is expected to be the first demonstration of such eco-evolutionary feedbacks in a 3D artificial creature model. We believe that our approach allows us to analyze the emergent process of various morphological and behavioral strategies in this context.

Conclusion

We presented the results of evolutionary experiments investigating the interaction between the population dynamics and the trait evolution of a predator-prey scenario in a 3D physically simulated environment. The morphologies and behaviors of virtual prey creatures are evolved using a genetic algorithm based on the predation interactions between predators and prey. We also changed the population size of both species depending on the fitness of individuals.

We found the different interactions between population and evolutionary dynamics at short and long timescales. When we focused on the short-term dynamics, we observed a simple cyclical dynamics of the population of predators and prey, which corresponds to a Lotka-Volterra type population dynamics. This is because the population dynamics was too fast for the evolutionary dynamics to adapt to. In contrast, when we focused on the long-term dynamics, we observed the complex interactions between the population dynamics of both species and the evolutionary dynamics of the trait of prey. Specifically, we found the inverse correlation between the population sizes and the average volume of the prey, and their continual fluctuations, yielding the emergence of defensive and non-defensive morphological strategies of prey. This is due to the fact that the target of selection for the prey switched between defense against predation and reduction in cost of defense depending on the population size of predators.

That is, the change in the population size caused the change in the selection pressure and the change in the trait caused the population change. We believe that such dynamics can be observed in predator-prey scenarios both in artificial frameworks and in nature.

Our model could be extended in various directions. One obvious direction would be to evolve the predators simultaneously. Such extended evolutionary experiments may show the population and evolutionary dynamics in the predator-prey relationship more clearly. Furthermore, we believe that the dynamics selection pressure exerted by an evolving predator would likely be a major factor in shaping the population and evolutionary dynamics, leading to more complex dynamics than the monotonous repetition of a similar evolution of the volume observed in this paper. Another direction would be to add intra-species interaction to support group hunting and prey herding behaviors and shed light on their

effect on the population dynamics.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number $26 \cdot 10516$.

References

- Agrawal, A. (2001). Phenotypic plasticity in the interactions and evolution of species. *Science*, 294(5541):321.
- Azarbadegan, A., Broz, F., and Nehaniv, C. L. (2011). Evolving sims's creatures for bipedal gait. In *Proceedings of 2011 IEEE Symposium on Artificial Life (IEEE ALIFE 2011)*, pages 218–224.
- Chaumont, N., Egli, R., and Adami, C. (2007). Evolving virtual creatures and catapults. *Artificial Life*, 13:139–157.
- Chen, J., Benesty, J., and Y., H. (2003). Time delay estimation using spatial correlation techniques. In *Proceedings of the 8th International Workshop Acoustic Echo and Noise Control (IWAENCO3)*, pages 207–210.
- Edmunds, M. (1974). Defence in Animals: A survey of antipredator defences. Longman Group Limited.
- Fussmann, G. F., Loreau, M., and Abrams, P. A. (2007). Ecoevolutionary dynamics of communities and ecosystems. *Functional Ecology*, 21(3):465–477.
- Hairston, N. G. J., Ellner, S. P., Geber, M. A., Yoshida, T., and Fox, J. A. (2005). Rapid evolution and the convergence of ecological and evolutionary time. *Ecology Letters*, 8(10):1114–1127.
- Hastings, A. (1997). Population Biology: Concepts and Models. Springer.
- Ito, T., Pilat, M. L., Suzuki, R., and Arita, T. (2012). Emergence of defensive strategies based on predator-prey coevolution in 3d physical simulation. In Proceedings of the 6th International Conference on Soft Computing and Intelligent Systems, and the 13th International Symposium on Advanced Intelligent Systems 2012 (SCIS-ISIS2012), pages 890–895.
- Ito, T., Pilat, M. L., Suzuki, R., and Arita, T. (2013a). Alife approach for body-behavior predator-prey coevolution: Body first or behavior first? Artificial Life and Robotics, 18(1-2):36–40.
- Ito, T., Pilat, M. L., Suzuki, R., and Arita, T. (2013b). Coevolutionary dynamics caused by asymmetries in predator-prey and morphology-behavior relationship. In *Proceedings of the 12th European Conference on Artificial Life*, pages 439–445.
- Kokko, H. and Lopez-Sepulcre, A. (2007). The ecogenetic link between demography and evolution: can we bridge the gap between theory and data? *Ecology Letters*, 10(9):773–782.
- Legreneur, P., Laurin, M., and Bels, V. (2012). Predator-prey interactions paradigm: a new tool for artificial intelligence. *Adaptive Behavior*, 20(1):3–9.
- Lotka, A. (1932). The growth of mixed populations: two species competing for common food supply. *Journal of the Washington Academy of Sciences*, 22:461–469.

- Miconi, T. (2008). In silicon no one can hear you scream: evolving fighting creatures. In *Proceedings of the 11th European conference on Genetic programming (EuroGP'08)*, pages 25–36.
- Miconi, T. and Channon, A. (2006). Analysing co-evolution among artificial 3d creatures. Artificial Evolution, 3871:167–178.
- Pilat, M. L., Ito, T., Suzuki, R., and Arita, T. (2012). Evolution of virtual creature foraging in a physical environment. In *Proceedings of the 13th International Conference on the Simulation and Synthesis of Living Systems (ALIFE13)*, pages 423–430
- Pilat, M. L. and Jacob, C. (2008). Creature academy: A system for virtual creature evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2008)*, pages 3289–3297.
- Pilat, M. L. and Jacob, C. (2010). Evolution of vision capabilities in embodied virtual creatures. In *Proceedings of the 12th an*nual Conference on Genetic and Evolutionary Computation Conference (GECCO 2010), pages 95–102.
- Post, D. M. and Palkovacs, E. P. (2009). Eco-evolutionary feedbacks in community and ecosystem ecology: interactions between the ecological theatre and the evolutionary play. *Philosophical Transcations of the Royal Society B*, 364(1523):1629–1640.
- Sanchez, A. and Gore, J. (2013). Feedback between population and evolutionary dynamics determines the fate of social microbial populations. *PLoS Biology*, 11(4).
- Schoener, T. W. (2011). The newest synthesis: Understanding the interplay of evolutionary and ecological dynamics. *Science*, 331(6016):426–429.
- Sims, K. (1994a). Evolving 3d morphology and behavior by competition. In *Proceedings of the 4th International Works on Synthesis and Simulation of Living Systems (ALIFE IV)*, pages 28–39.
- Sims, K. (1994b). Evolving virtual creatures. In 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 94), pages 15–22.
- Steven, A. (1998). *Foundations of Social Evolution*. Princeton University Press.
- Taylor, T. and Massey, C. (2000). Recent developments in the evolution of morphologies and controllers for physically simulated creatures. *Artificial Life*, 7(1):77–87.
- Vorterra, V. (1926). Fluctuations in the abundance of a species considered mathematically. *Nature*, 118:558–560.
- Yoshida, T., Ellner, S. P., and Hairston, N. G. J. (2004). Evolutionary tradeoff between defense against grazing and competitive ability in a simple unicellular alga, chlorella vulgaris. In Proceeding of the Royal Society of London Series B-Biological Sciences, volume 271, pages 1947–1953.
- Yoshida, T., Ellner, S. P., Jones, L. E., Bohannan, B. J. M., Lenski, R. E., and Hairston, N. G. J. (2007). Cryptic population dynamics: rapid evolution masks trophic interactions. *PLoS Biology*, 5:1868–1879.

Building on Simplicity: Multi-stage Evolution of Digital Organisms

Laura M. Grabowski, Javier A. Magaña

Department of Computer Science, The University of Texas–Pan American 1201 W. University Drive, Edinburg, Texas, 78539 grabowskilm@utpa.edu

Abstract

The issue of how complex organismal features and functions arise through evolution remains a topic of great interest to researchers. We used digital organisms to investigate how simpler capabilities that evolved earlier may provide stepping stones for evolving new features. Populations of digital organisms evolved in environments of varying complexity where survival required different uses of memory. We conducted experiments that used different initial ancestors for populations: (1) an ancestor with only the ability to selfreplicate, evolved in each of the four types of experimental environments; (2) successful organisms from one environment seeded populations that continued to evolve in one additional environment ("one level" transfer); (3) successful organisms were transferred progressively from the simplest environment to the most complex ("waterfall" transfer). We found that the difference in performance of the most successful organisms at the end of evolution was significantly different (1) when the original environment differed little from the new environment, and (2) when the original environment produced adaptations that provided memory and use of previous life experience.

Introduction

In order to survive, organisms have evolved many complex adaptations. Even simple organisms have complex functions and structures. Bacteria communicate using secreted molecules to coordinate the behavior of the group. In other organisms, complex hierarchical regulatory circuits have evolved to integrate and process sensory information (Brelles-Marino and Bedmar, 2001). Many complex adaptations gave rise to brains that are capable of extracting patterns from a noisy, non-stationary, and often unpredictable environment. Brains control and coordinate movement, form memories, and construct models of the world and its dynamics (Koch and Laurent, 1999).

The manner in which complex features such as brains evolve is a key issue in evolutionary biology. Insight into this issue helps explain the presence of biological complexity and provides inspiration to computer scientists and engineers. Investigating the evolution of complexity can provide guidance in how we create and use new technologies, including drug therapies, the Internet, and self-replicating robots.

Trying to find effective ways to improve on these technologies is challenging, but identifying how basic elements affect complexity may lead to new and more effective solutions to complex problems.

Coyne (2009) describes natural selection as a "tinkerer," doing the best it can with what is available at the time. In the natural world, evolution through natural selection has repeatedly produced highly complex traits. To understand the details of how evolution produces something as complex as a brain, we would need a perfect fossil record, ideally including the genetic information for the entire line of descent. This ideal situation rarely exists in the natural world, making such analysis hard to perform in living organisms. Digital evolution, an approach within the larger discipline of artificial life, offers a potential solution to many of the problems involved with studying evolution in living organisms. Digital evolution systems allow us to track evolution as it occurs, producing the complete record we need for analysis. Researchers can observe and analyze the evolving population without disturbing the environment, something that is nearly impossible to do in natural systems. Investigators are also able to control most aspects of the evolutionary environment so that hypotheses can be tested with proper experimental control. This degree of control is difficult to achieve with natural systems.

Many researchers have explored issues of complexity. The notion of complexity can be viewed in a number of ways. These different perspectives are reflected in the variety of approaches to exploring the topic. Adami et al. (2000) measured genomic complexity in digital organisms using information theory. Adami et al. showed that genomic complexity increases under the influence of natural selection, as mutations that allow organisms to survive (i.e., informative mutations) are preserved in the genome. Goldstein (2009) applied fluid dynamics principles to the question of the evolution of multicellularity, hypothesizing that the transition to multicellularity was connected to competing aspects of fluid dynamics. Lenski et al. (2003) provided a strong demonstration of the emergence of functional complexity, the degree of complexity of the realized function.

Their study showed that there are functional relations between simpler and more complex functions, and that simpler functions may be used to evolve more complex functions. In that study, Avida organisms evolved to perform nine logic functions of increasing complexity. Of the 50 experimental populations, 23 populations evolved the most complex task, logical equals (EQU). In another set of experiments for the study, organisms were rewarded for performing EQU, but not for performing any of the simpler tasks. In this setup, none of the populations evolved EQU. The authors' analysis revealed that the complex task required building blocks provided by the simpler tasks that came earlier in evolution.

In this paper, we present results related to the evolution of functional complexity. Digital organisms evolved in different environments where success required different uses of memory. The complexity of the task related to the level of memory use required to execute the task. To test the hypothesis that simpler functions that evolved earlier provided building blocks for more complex functions, we used three treatments with different initial ancestors for the evolving populations.

Avida Overview

The Avida Digital Evolution platform (Ofria and Wilke, 2004) places a population of self-replicating computer programs (called digital organisms or Avidians) in a computational environment. Avida provides the basic ingredients necessary for evolution: replication, competition and variation (Dennett, 2002).

The Avida world, or population grid, is a two-dimensional grid of cells, each of which may contain at most one digital organism. Each individual organism is made of a circular list of assembly language-like instructions, its "genome," and a virtual central processing unit (CPU). The standard Avida virtual CPU consists of three registers, two stacks, and several heads. Each instruction in the genome modifies the virtual CPU, and the cost of executing the instructions is measured in virtual CPU cycles. Avidians perform all internal operations and interact with the world through executing their instructions.

Most Avida experiments are seeded with a simple self-replicating ancestor, *i.e.*, an organism that has only the ability to copy itself into a space of memory that will become its offspring. When the digital organisms copy themselves (*i.e.*, self-replicate), there is a possibility for variations, or mutations. Mutations in Avida can be an addition, deletion, or change of Avida instructions in the offspring's genome. When an organism has finished its replication, the offspring's genome is divided from that of the parent, and the offspring is placed in a grid cell in the population grid, replacing any other organism that occupied that location. Thus, the fundamental competition in Avida is created by the limited space in the environment. An Avidian that can replicate faster has an advantage over slower replicators because

the faster replicating organism will have a higher probability of more descendants in future populations than a slower Avidian. Execution speed is variable, and is determined in part by an organism's metabolic rate. The metabolic rate is used to allocate virtual CPU cycles, and a higher metabolic rate increases the speed the Avidian can execute instructions by allocating it more virtual CPU cycles in the current time slice. Avidians may increase their metabolic rate by performing tasks related to success, such as performing logic operations or following a path.

In our experiments, we use an Avida environment type called the state grid (Grabowski et al., 2010, 2011, 2012, 2013). The state grid, or behavior grid, is a virtual environment separate from the Avida population grid, where each organism has separate information about its environment. While a digital organism stays in the same position in the Avida population grid, each Avidian has a virtual grid where it can move independently of other organisms in the population. The only interaction between individuals in behavior grid experiments is when a newly produced offspring replaces another organism in the population grid. The behavior grid allows simplification of the experimental design and implementation of experiments, is easily defined and understood by human experimenters, and runs with efficiency and relatively low computational overhead.

Methods

Evolutionary environments

Each behavior grid contains a single path for the Avidians to follow. The path is formed by placing a sensory cue in each grid cell. Avidians may sense the cue in the cell if they execute a sensing instruction. The cues do not provide any information to the organisms about what to do with the sensory information; organisms must decide what use to make of the sensory cue, if any. The concept behind the environments is that an organism moves around the behavior grid environment, encountering different states that either increase or decrease the organism's energy by increasing or decreasing the metabolic rate bonus. When an organism moves along the path, it encounters "food" states that supply more energy than the amount of energy lost through movement. If an organism moves to a location off the path, it will not receive energy since those cells do not contain food. The more steps the organism takes off the path, the greater the amount of energy wasted, reducing the overall metabolic rate bonus. Organisms that move along the food path build up energy, and are able to execute at an accelerated rate.

The environmental cues or signals in the behavior grid paths vary depending on the experimental design. We used the following cues to construct behavior grid paths:

• *Empty*: indicates that the organism is off the path. Movement in empty cells reduces metabolic rate bonus.

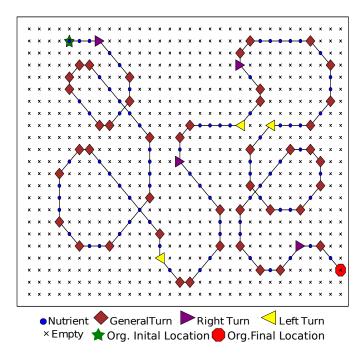


Figure 1: Example environment for the General-turn (GT) experiments. GT environments contain all sensory cues used to construct behavior grid paths.

- *Nutrient*: indicates that a cell is on the path and contributes to the metabolic rate bonus the first time that the organism occupies the cell.
- Directional cue: signals a turn (either right or left) in the path. All turns are 45° increments. Turn cues are treated as nutrients, adding to the metabolic rate bonus.
- General-turn cue: signals the repetition of the most recently encountered directional cue and also contributes to the metabolic bonus.

Different environments are created by using different cues for the paths. The configurations of our paths were inspired by maze-following experiments with honey bees (Zhang et al., 1996). For our experiments, we used the following environment types:

- *Single-turn (ST)*: paths that have directional cues in only one direction, right or left.
- Right-left (RL): environments containing both left and right directional cues in the same environment.
- Cue-first (CF): the first turn signal (i.e., right or left directional cue) in the environment specifies the direction of the rest of the turns, which are indicated by general turn cues. All turns in each environment are in the same direction (right or left).

• General-turn (GT): contains a combination of left and right directional cues, along with the general turn cue. The explicit directional cue appears when the direction of turn changes from the preceding turn (e.g., when the last turn was to the right and the current turn is to the left). Otherwise, the general turn cue appears. Figure 1 shows an example GT environment.

Each environment is slightly more complex than the previous one in terms of the memory needed for success in the environment. In ST environments, Avidians must handle only one directional cue type in each environment. For RL environments, organisms must respond to both types of directional cues. The ST and RL environments require only a simple mapping from the sensory input to the necessary action. The CF and GT environments, however, require that organisms make behavioral decisions based on both the current sensory input and past life experience. In CF environments, that experience is the specific directional cue that was sensed at the first turn on the path. Once the organism successfully maps the general turn to that initial explicit directional cue, it does not need to update that mapping during its lifetime. For GT environments, the organism must remember the most recent directional cue. GT environments require the ability to remember the last directional cue and update that memory at irregular intervals while traversing the path. A successful solution for the GT environment is also a successful solution for all the other environments. Each environment is a sub-problem of the subsequent environment.

Organisms receive a metabolic rate bonus for a path traversal task (Grabowski et al., 2010), accruing increasing bonuses as they move farther on the path. If an organism travels the entire path without stepping off, it receives the maximum possible bonus. Organisms receive the bonus only for the first time they enter a path cell, discouraging them from evolving to oscillate between cells. Any movement off the path is considered a waste of energy, and therefore all movement off the path reduces metabolic bonus.

Task quality (TQ) provides a measure of how well an organism traverses a path, reflecting the proportion of the path an organism travelled without making mistakes. The calculation of task quality is simply

$$TQ = ((valid - empty)/pathLength)$$

where (valid-empty) is the difference of the count of unique (i.e., non-duplicate) path cells encountered (valid) and the total count of empty cells (empty) encountered, and pathLength is the total count of path cells (nutrients and directional cues). Task quality must remain non-negative. If valid-empty becomes negative, task quality is set to 0. Task quality values range from 0 to 1, with 1 the best possible task quality value. Avida tracks the task quality throughout evolution, recording the maximum task quality (MTQ). The MTQ value is the highest task quality among all organisms in the population at the moment the measurement

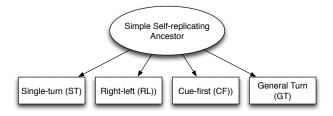


Figure 2: Control experiments. Populations in all experimental environments were seeded with the simple self-replicating ancestor (a digital organism with only the ability to replicate).

is recorded. To provide aggregate task quality information across all populations, we use the average maximum task quality (AMTQ). AMTQ takes the average of the MTQ values for all populations at each recorded time point, allowing us to look at what happened to the MTQ values throughout evolution.

Experimental Setup

We used three treatments for experiments, one which served as a control and two experimental treatments. For the control (Figure 2), we seeded populations in each of the four environment types described above (ST, RL, CF, GT) with an organism with only the ability to replicate. The control populations established a baseline against which to compare the experimental populations, and replicated results of earlier work (Grabowski et al., 2010, 2011, 2012). For each type of environment, we used eight different path configurations. Each organism was assigned a path at random when it was placed in the population grid. All experiments ran for 250,000 updates (the native unit of time in Avida, equivalent to around 10,000 generations in these experiments).

The purpose of the experimental treatments was to probe how earlier adaptations affect further evolution. For these experiments, organisms that evolved in one environment were transferred to, and continued to involve in, a different environment. For both treatments described below, we ranked the populations by maximum task quality at the end of evolution and took the best organism from each of the top five populations. Each of these five organisms served as the initial ancestor for ten new populations in the new environment. Experimental treatments differed according to the number of different environments organisms were exposed to during evolution, as described below.

1. **Experiment 1: "One level" transfer.** The top five organisms from each of the first three environments in the control experiments (ST, RL, CF) were transferred to the next most difficult environment. The one level transfers are illustrated in Figure 3.

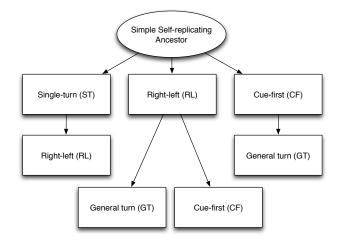


Figure 3: One level transfer experiments. Populations in all experimental environments were seeded with organisms that had evolved in one prior environment.

2. Experiment 2: "Waterfall" transfer. In this treatment, evolved organisms were transferred progressively from the simplest environment (ST) through the most complex environment (GT). Evolved organisms from the ST control experiments served as the ancestors for the RL environment populations, as described above (see Experimental Setup). After evolution in this new environment, the most successful organisms became the ancestors for populations in both the CF and GT environments. After additional evolution in the CF environment, successful organisms seeded more GT environment populations (Figure 4).

In both experimental treatments, we used two different transfer "pathways" into GT environments (RL to GT, and CF to GT). This approach allowed us to test which sub-problem—and its corresponding previously evolved ability—bestowed more of an advantage, the ability to handle both right and left turns in the same path or the ability to remember a past action.

Results and Discussion

One Level Transfer Experiments

As described above, the ancestor organisms for these experiments were the highest fitness organisms from the top five control populations, ranked by task quality. Each of the five organisms was the ancestor for ten new populations in the transfer environment. Not all of the new populations survived the transfer, *i.e.*, some populations died out before the planned end of the experiment. Table 1 summarizes MTQ and survival information for the one level transfer experiments.

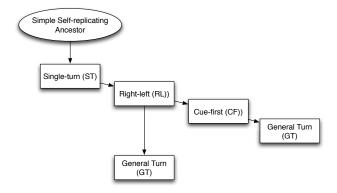


Figure 4: Waterfall transfer experiments. Top organisms from the control experiments in the ST environment seeded new populations that evolved further in RL environments. Organisms from the most successful of those populations then seeded new populations in CF and GT environments. After more evolution in the CF environment, organisms from top performing populations seeded new populations in the GT environment.

Single-turn to Right-left environments (ST-RL) The top five ranking organisms from ST environments were transferred to RL environments. Since these organisms' ancestors were exposed to both right and left turn paths during the course of evolution in the ST environment, we hypothesized that the transfer populations would be able to evolve a solution faster than the control populations that evolved from scratch (de novo). Our data support this hypothesis. In fact, the transfer populations evolved the capability to traverse the entire path correctly in a short time, and maintained that ability throughout the duration of the experiments. The difference in maximum task quality (MTQ) values between the transfer populations and the populations evolved de novo is statistically significant (Mann-Whitney, p < 0.05). Figure 5 shows the distributions of the MTQ values at the end of evolution for both treatments, illustrating that all the ST-RL transfer populations have TQ values very near 1.0. The ST control populations have a high mean, but more variation in the overall distribution of TQ values.

Right-left to Cue-first environments (RL-CF). In the the Right-left to Cue-first (RL-CF) transfer experiments, only two of the five seed organisms survived the transfer to the new environment. Thus, only 20 of 50 populations survived to the end of the experiment. All of the surviving populations come from the third and fifth ranked organisms transferred from the RL environment.

Figure 6 shows that the AMTQ of the transfer populations was higher than that of the control populations at the beginning of the experiments. As evolution progressed, the AMTQ of the control populations passed that of the transfer

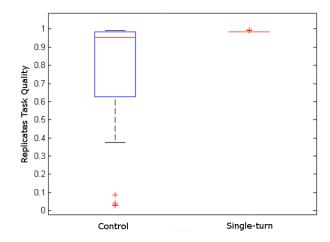


Figure 5: One level transfer experiments, Single-turn to Right-left environments (ST-RL) and Control. Top organisms from the control experiments in the ST environment seeded new populations that evolved further in RL environments. Plot shows the distribution of maximum task quality (MTQ) values for all replicate populations at the end of evolution for the two treatments. The difference in the MTQ values is significant (Mann-Whitney, p < 0.05).

populations. Although some of the individual transfer populations did well, in general they did not fare as well as the control populations that evolved from scratch in the CF environments. Of all of the transfer populations that survived to experiment completion, only two produced organisms that were able to successfully traverse the Cue-first environment paths. Both of these organisms evolved from the fifth-ranked transfer ancestor.

Right-left to General-turn environments (RL-GT). The results of the Right-left to General-turn (RL-GT) transfer experiments echo those of the RL-CF experiments. Only 20 of 50 populations survived the experiment, with ten of the surviving populations descended from the third-ranked seed organism, and the other ten from the fifth-ranked seed organism. Figure 7 shows the AMTQ values for all treatments for GT environments (control populations, RL-CF transfers, and experiments discussed in the next two sections, CF-GT transfers, and waterfall transfers). The AMTQ of the GT and the RL-GT experiments are very close to each other. At the beginning of evolution, the transfer populations perform better than the controls. As evolution continues, the control populations surpass the AMTQ of the transfer populations. The transfer populations appear to evolve some improvements at intervals, so that the AMTQ of the transfer populations approaches that of the control populations. The populations that evolved from scratch steadily increase performance over time, but are surpassed by the transfer pop-

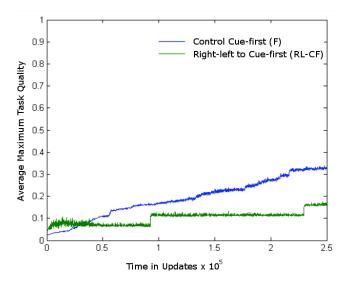


Figure 6: One level transfer experiments, Right-left to Cuefirst environments (RL-CF), and Control. Top organisms from the control experiments in the RL environment seeded new populations that evolved further in CF environments. Transfer populations initially had higher task quality than the control populations, but failed to make much progress over evolution. The control experiments continued to improve slowly throughout the course of evolution, ending the experiment with a significantly higher AMTQ than the transfer populations (Mann-Whitney, p < 0.05).

ulations just before the experiment finishes. However, the difference in AMTQ at the end of experiments is not significant (Mann-Whitney, p>0.05). The individual organism with the greatest task quality was from a RL-GT transfer population, and had an AMTQ of 0.9926, indicating near perfect traversal of the path. Figure 8 shows the distribution of MTQ values for the RL-GT populations and the control GT populations.

Cue-first to General-turn environments (CF-GT). Some of the most interesting results came from the Cue-first to General-turn transfer experiments (CF-GT). 28 of 50 populations survived the experiment, with surviving populations descended from the first, second and fourth ranked seed organisms.

Figure 7 shows that the CF-GT produced the best AMTQ among the GT experiments for all treatments discussed so far. The transfer populations performed better than populations from other treatments from the start, reaching an AMTQ of 0.5 before the midpoint of the experiment, with a steady increase throughout the remainder of evolution. By the end of the experiment the populations had an AMTQ greater than 0.6. Figure 8 shows the MTQ distribution from the GT, the RL-GT, and CF-GT experiments. There is a significant statistical difference between the CF-GT

Experiment	Surviving Replicates	Replicates with MTQ > 0.9
ST-RL	40	40
RL-CF	20	2
RL-GT	20	4
CF-GT	28	7

Table 1: Summary of one level transfer experiments. "Experiment" identifies the evolutionary environment of the ancestor and the environment to which the evolved organisms were transferred. For example, "ST-RL" denotes that the ancestor evolved first in the ST (Single-turn) environment and was then transferred to the RL (Right-left) environment. "Surviving Replicates" is the total number of replicates (populations) that survived until the end of the experiment in the new environment. "Replicates with MTQ > 0.9" gives the number of surviving populations that produced a Maximum Task Quality (MTQ) above 0.9 at the end of evolution.

and the other two groups, GT and RL-GT (Kruskal-Wallis, $p < 0.05). \label{eq:groups}$

Waterfall Transfer Experiments

The waterfall transfer experiments begin with the control experiments evolving in ST environments. The top five organisms from the ST control experiment are then used to seed populations in RL environments. The top five organisms from these transfer populations are used to seed populations in both CF and GT environments. Finally, the top five organisms from the CF transfer populations seed new populations in GT environments. In the one level transfer experiments discussed above, many populations did not survive the transfer to the new environment. Survival was not an issue, however, in the waterfall transfer experiments, since all the populations survived to the end of the experiments.

ST-RL-GT Waterfall Transfer. The ST-RL-GT waterfall experiments produced the lowest AMTQ among all the treatments discussed so far. Figure 7 shows the AMTQ for GT populations from all treatments. During the first half of the experiment, the performance of the populations is similar to that of the populations from RL-GT one level transfer experiments. After that time, improvement appears to level off. At the end the experiment, these populations have the lowest AMTQ among all treatments. Figure 8 shows the MTQ distribution for all treatments in the GT environment. The difference between between AMTQ values for the treatments is statistically significant (Kruskal-Wallis, p < 0.5).

ST-RL-CF-GT Waterfall Transfer. The ST-RL-CF-GT waterfall experiment has the second highest AMTQ values among all treatments in the GT environment (Figure 7). The ST-RL-CF-GT populations increase their AMTQ quickly at

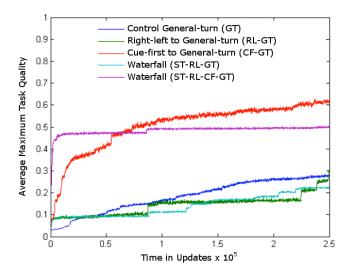


Figure 7: General-turn environment (GT), all treatments: ST-RL-GT waterfall experiments, ST-RL-CF-GT waterfall experiments, one level transfer experiments (RT-GT and CF-GT) and Control. The plot shows that the transferred CF-GT organism has the best AMTQ, while the ST-RL-GT has the lowest AMTQ among all treatments. The difference in AMTQ values between treatments was significant (Kruskal-Wallis test, p < 0.05).

the start of evolution, reaching an AMTQ above 0.4 with more than 90% of the experiment's time remaining. After the impressive start, however, improvement levels off, ending the experiment with an AMTQ value near 0.5 at the end of the experiment. Figure 8 shows the MTQ distributions from all treatments in the GT environment (Control, RL-GT, CF-GT, ST-RL-GT, and ST-RL-CF-GT). The difference between the MTQ distributions is significant (Kruskal-Wallis, p < 0.05). Table 2 gives a summary of the waterfall transfer experiments, showing the waterfall path (ST-RL-GT, or ST-RL-CF-GT) and the number of replicates that had MTQ value above 0.9. Similar to the one level transfer populations, the waterfall transfers from the CF environment had better MTQ, and a higher number of populations with MTQ value above 0.9.

Experiment	Replicates with MTQ > 0.9
ST-RL-GT	6
ST-RL-CF-GT	23

Table 2: Summary of waterfall transfer experiments, showing the waterfall path (ST-RL-GT, or ST-RL-CF-GT) and the number of replicates that had MTQ greater than 0.9. All 50 populations in each pathway survived to the end of the experiments in these treatments.

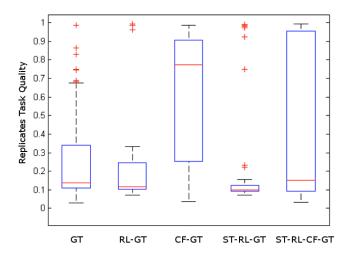


Figure 8: General-turn environment (GT), all treatments: ST-RL-GT waterfall experiments, ST-RL-CF-GT waterfall experiments, one level transfer experiments (RT-GT and CF-GT) and Control. Plot showing the distribution of MTQ values at the end of evolution for the different treatments. The difference in the MTQ values is significant (Kruskal-Wallis, p < 0.05).

Conclusions and Future Work

Our results provide mixed support for our hypothesis that earlier evolution will provide building blocks for evolving more complex functions. We observed strong effects of advantages bestowed by prior evolution in the one level ST-RL and CF-GT transfer experiments. From one perspective, these results are not surprising. Intuitively, some environments equipped organisms with more stepping stones toward the new environment than others. Organisms that evolved in the ST environment were exposed to both right and left turn environments over the course of evolution. Successful organisms would be able to react to both directional cues, providing an important advantage when transferred to RL environments. The critical ability that evolved in the CF environments was memory, the same fundamental capability that was needed for success in the GT environments. If organisms were capable of remembering past experience from the start of evolution in GT environments, evolution only needed to tinker with the use of memory to suit the details of the new environment. The RL environment did not provide this critical stepping stone required for success in the CF or GT environment. Success in RL environments required that organisms correctly recognize and react to the directional cues, with no memory needed. Thus, organisms that evolved initially in RL environments lacked the key memory ability needed for either the CF or GT environment.

We were initially somewhat puzzled by the lack of advan-

tage provided by earlier evolution in the waterfall transfer experiments. We suggest some possible explanations for the results. First, perhaps the other environments are too simple, and the differences in demands between environments are not pronounced enough to produce a significant effect. A second explanation is that the solutions that evolved earlier were not easily modified by more evolution. In other words, perhaps the solutions were not sufficiently evolvable. Both the lackluster performance of the transfer populations and the high mortality rate of one level transfer populations suggest that low evolvability may be an issue. The noticeably bad performance of the ST-RL-GT transfer populations reinforces this possible explanation. We have several avenues in mind for exploring this issue. We will do additional analysis on evolved genomes for evidence of overspecialization, such as high redundancy or large genomes. We will also analyze the structure and algorithms of selected evolved genomes to identify features that are tuned to a specific environment. Such tuning may be evidence for over-specialization. Experiments of long duration may also lead to a high degree of specialization or other factors that impact evolvability. For all three treatments, experiments were given a relatively long run time (250,000 Avida updates, around 10,000 generations on average). That length of time is more than enough for a particular solution to tune to the initial environment to such a degree that transferred organisms cannot survive the new environment. We plan to test this hypothesis with another series of experiments with shorter durations before transfers occur.

Another possible explanation for the waterfall transfer results is a bottleneck effect from using only the top five individuals as seeds for new populations. In addition to the loss of genetic diversity through the bottleneck, effects of random drift may be amplified. To counter these issues, we will do new experiments in which we seed new populations with an entire population from a prior environment.

The difference in survival rates between one level and waterfall transfers was also interesting. We plan to repeat our experiments while controlling the seed values used by Avida at the start of experiments, so that the waterfall transfer experiments use the same seed values as the one level transfer experiments. By doing this, we can test for whether the poor performance in the waterfall transfer experiments is related to bad luck in random seeds, or a more substantive effect.

We would also like to take the evolved genomes from the various experiments and analyze them in terms of information content, to provide a more quantitative view of the information required for these environments. Such analysis will help us to design improved environments and measurements for exploring the evolution of functional complexity.

References

Adami, C., Ofria, C. A., and Collier, T. C. (2000). Evolution of biological complexity. *Proceedings of the National*

- Academy of Science, 97:4463-4468.
- Brelles-Marino, G. and Bedmar, E. J. (2001). Detection, purification and characterisation of quorum-sensing signal molecules in plant-associated bacteria. *Journal of Biotechnology*, 91(2–3):197–209.
- Coyne, J. A. (2009). Why Evolution Is True. Viking, New York.
- Dennett, D. C. (2002). The new replicators. In Pagel, M., editor, *Encyclopedia of Evolution*, pages E83–E92. Oxford University Press, New York.
- Goldstein, R. (2009). Evolution of biological complexity. In *Séminaire Poincaré XII*, pages 75–88. Institut Henri Poincaré.
- Grabowski, L. M., Bryson, D. M., Dyer, F. C., Ofria, C., and Pennock, R. T. (2010). Early evolution of memory usage in digital organisms. In Artificial Life XII: Proceedings of the Twelfth International Conference on the Synthesis and Simulation of Living Systems, pages 224–231, Cambridge, MA. MIT Press.
- Grabowski, L. M., Bryson, D. M., Dyer, F. C., Pennock, R. T., and Ofria, C. (2011). Clever creatures: case studies of evolved digital organisms. In *Proceedings* of the Eleventh European Conference on the Synthesis and Simulation of Living Systems (ECAL 2011), pages 276–283, Cambridge, MA. MIT Press.
- Grabowski, L. M., Bryson, D. M., Dyer, F. C., Pennock, R. T., and Ofria, C. (2012). An analysis of the de novo evolution of a complex odometric behavior. In *Proceedings of the 13th International Conference on the Simulation and Synthesis of Living Systems (ALife 13)*, pages 585–586, Cambridge, MA. MIT Press.
- Grabowski, L. M., Bryson, D. M., Dyer, F. C., Pennock, R. T., and Ofria, C. (2013). A case study of the de novo evolution of a complex odometric behavior in digital organisms. *PLOS ONE*, 8(2772):e60466.
- Koch, C. and Laurent, G. (1999). Complexity and the nervous system. *Science*, 284(5411):96–98.
- Lenski, R. E., Ofria, C., Pennock, R. T., and Adami, C. (2003). The evolutionary origin of complex features. *Nature*, 423:139–144.
- Ofria, C. and Wilke, C. O. (2004). Avida: a software platform for research in computational evolutionary biology. *Artificial Life*, 10(2):191–229.
- Zhang, S. W., Bartsch, K., and Srinivasan, M. V. (1996). Maze learning by honeybees. *Neurobiology of Learning and Memory*, 66:267–282.

More Bang For Your Buck: Quorum-Sensing Capabilities Improve the Efficacy of Suicidal Altruism

Anya Elaine Johnson^{1,5}, Eli Strauss^{2,5}, Rodney Pickett^{1,5}, Christoph Adami^{3,5}, Ian Dworkin^{2,5}, & Heather J. Goldsby^{4,5}

¹Computer Science and Engineering, Michigan State University, East Lansing, MI, USA
²Zoology, Michigan State University, East Lansing, MI, USA
³Microbiology and Molecular Genetics / Physics and Astronomy, Michigan State University, East Lansing, MI, USA
⁴Biology, University of Washington, Seattle, WA, USA, ⁵BEACON Center for the Study of Evolution in Action anyaejo@msu.edu

Abstract

Within the context of evolution, an altruistic act that benefits the receiving individual at the expense of the acting individual is a puzzling phenomenon. An extreme form of altruism can be found in colicinogenic E. coli. These suicidal altruists explode, releasing colicins that kill unrelated individuals, which are not colicin resistant. By committing suicide, the altruist makes it more likely that its kin will have less competition. The benefits of this strategy rely on the number of competitors and kin nearby. If the organism explodes at an inopportune time, the suicidal act may not harm any competitors. Communication could enable organisms to act altruistically when environmental conditions suggest that that strategy would be most beneficial. Quorum sensing is a form of communication in which bacteria produce a protein and gauge the amount of that protein around them. Quorum sensing is one means by which bacteria sense the biotic factors around them and determine when to produce products, such as antibiotics, that influence competition. Suicidal altruists could use quorum sensing to determine when exploding is most beneficial, but it is challenging to study the selective forces at work in microbes. To address these challenges, we use digital evolution (a form of experimental evolution that uses self-replicating computer programs as organisms) to investigate the effects of enabling altruistic organisms to communicate via quorum sensing. We found that quorumsensing altruists killed a greater number of competitors per explosion, winning competitions against non-communicative altruists. These findings indicate that quorum sensing could increase the beneficial effect of altruism and the suite of conditions under which it will evolve.

Introduction

An organism behaves altruistically when it performs an action that lowers its own fitness in order to increase the fitness of another organism (West et al., 2007). Altruistic organisms ranging from bacteria to humans are found frequently in nature (Chao and Levin, 1981; Bowles, 2006). We might expect that any altruistic genes should be selected against because the altruistic organism will have a lower fitness than a non-altruistic organism benefitting from an altruistic act (a "cheater"). A particularly challenging form of altruism to explain is suicidal altruism, where the altruistic organism dies to increase the fitness of other organisms (Velicer

et al., 2000; Khare et al., 2009). However, inclusive fitness theory and multi-level selection theory both describe conditions under which altruism—even suicidal altruism—can evolve (Traulsen, 2006; Hamilton, 1963). According to inclusive fitness theory, altruism is favored by selection when an organism's altruistic action directly benefits its kin, who are likely to share the same altruistic gene(s) (Khare et al., 2009). However, an open question is how do altruistic organisms know when to behave altruistically? For example, how do altruistic organisms know when their kin surround them and would benefit from their actions?

Under certain conditions, colicinogenic E. coli commit suicide to kill nearby competitors and thus free resources for their kin (Chao and Levin, 1981). Specifically, these E. coli stochastically produce a toxin until they explode. Once an E. coli explodes, the toxin is released, killing all surrounding organisms that are not resistant. Because the exploding organism's kin are usually resistant to the toxin, they survive the explosion and benefit from the decrease in surrounding competition for resources (Chao and Levin, 1981). However, the prevailing environmental conditions, such as the number of competitors and kin surrounding the altruistic organism, determine the benefits of the suicidal action. Specifically, when an organism explodes, it is possible that either competitors are not present (and thus the explosion does not affect the competition for resources) or kin are not nearby (and thus are unable to take advantage of the accessible resources). In both of these cases, the altruistic organism committed suicide without providing any benefit to its kin.

An organism that can communicate with constituents to gain information about its environment could potentially improve decisions regarding when to perform altruistic actions. Quorum sensing is a type of communication found in many species of bacteria (Diggle et al., 2007; Davies, 1998), whereby the bacteria can assess how many constituent organisms surround them. Quorum sensing involves an organism releasing a small amount of a signaling protein and gauging the amount of that same protein that has been released by other bacteria around it (Bassler, 2002). A number of organisms combine quorum sensing with antibiotic

production to strategically kill competitors (Chandler et al., 2012). In these situations, producing an antibiotic is an expensive action and requires the cooperation of other constituents to produce enough to harm competitors. If too few organisms are producing the antibiotic, the action is expensive and does not accrue any benefits. However, if the organisms achieve quorum and ensure that enough constituents are also producing the antibiotic, then it is likely they can harm or kill the competitor. Thus, quorum sensing is used to enable organisms to determine whether the environmental conditions are favorable to perform expensive and possibly altruistic acts.

In this study, we explore whether altruistic organisms that do not have access to information about their environment would benefit from quorum-sensing capabilities. While enabling colicinogenic E. coli to use quorum sensing would determine the benefit of such a strategy, the challenges involved with performing that experiment are numerous. They include the difficulty associated with evolving complex traits, such as quorum sensing and suicidal altruism, and the generation time of E. coli. To address these experimental challenges, we use Avida, a digital evolution system (Ofria and Wilke, 2004). Within Avida, computer programs ("digital organisms") self-replicate with a chance of mutation and compete for space within their environment. Avida has been used to study evolutionary topics such as division of labor (Goldsby et al., 2012) and the evolution of biological complexity (Lenski et al., 2003). Additionally, Avida has been used to study suicidal altruism (Goings et al., 2004) and quorum sensing (Beckmann and McKinley, 2009) in isolation, making it the ideal platform for evolution experiments that combine these two complex traits. In this study, we use digital organisms that explode based on stochastic factors and quorum-sensing information when available. When an organism explodes, it kills competitors, and thus behaves similarly to colicinogenic E. coli. We study whether such organisms will evolve to use quorum-sensing capabilities and whether these capabilities provide a competitive advantage against altruists who cannot communicate.

Related Work

Suicidal altruism and communication have been studied extensively in both organic and digital systems (Berngruber et al., 2013; Bordereau et al., 1997; Chao and Levin, 1981; Crespi, 2001; Davies, 1998; Diggle et al., 2007; Foster et al., 2006; Goings et al., 2004; Goldsby et al., 2012; Hamilton, 1964, 1963; Kerr et al., 2004). Using both types of systems, researchers have found strong evidence that the benefits from higher inclusive fitness enables the success of suicidal-altruism strategies (Chao and Levin, 1981; Goings et al., 2004; Berngruber et al., 2013; Bordereau et al., 1997; Crespi, 2001; Foster et al., 2006). Furthermore, studies in both organic and digital systems have found that cooperation mediated by quorum sensing can be a successful strategy in a

number of environments (Diggle et al., 2007; Davies, 1998). Here we discuss relevant studies surrounding the evolution of suicidal altruism and quorum sensing within organic and digital systems.

Within nature, suicidal altruistic acts are generally performed to increase the success of the altruistic organism's kin (Chao and Levin, 1981; Berngruber et al., 2013). For example, colicinogenic *E. coli* stochastically kill competitors in an explosion of toxins, thus the surviving colony members have less competition for resources (Chao and Levin, 1981). Additionally, in some species of *E. coli*, when a bacterium of *E. coli* carrying the suicide gene *Lit* is infected by a lytic phage, it may kill itself to prevent the spread of the pathogen to the rest of the colony (Berngruber et al., 2013).

There are a number of organisms that use quorum sensing to regulate altruistic behavior (Miller and Bassler, 2001; Dworkin and Kaiser, 1985; Chandler et al., 2012). For example, Myxococus xanthus is a bacterium that uses quorum sensing to detect cell density. When confronted with starvation, at high cell density this bacterium forms fruiting bodies (Miller and Bassler, 2001). For the fruiting body to be successful, the majority of organisms must lyse to form the stalk and only a small fraction of the organisms reproduce as spores (Dworkin and Kaiser, 1985). Additionally, a number of species of bacteria use quorum sensing to kill competitors with antibiotics (Chandler et al., 2012). Competitors only die once the amount of antibiotics in the environment reaches a certain concentration (Chandler et al., 2012). If the antibiotic that is produced is too diluted, competitors may become resistant to it. As such, the bacteria are most likely to succeed in harming the competitors if a group of them simultaneously produce the antibiotic. Quorum sensing is used to establish that such a group is present and available to produce the antibiotic.

Altruism and quorum sensing have previously been studied in digital systems as well. Goings et al. studied suicidal altruism modeled on the colicinogenic E. coli's behavior (Goings et al., 2004). They discovered that suicidal altruism could evolve in the Avida digital evolution platform, even though the organisms could not communicate with each other. Suicidal altruism was most likely to evolve in a structured environment, where an organism had a high probability of being surrounded by its kin, rather than in a well-mixed environment, where an organism's kin were distributed randomly. In a subsequent study investigating environmental factors that influence the evolution of altruism, we found that large population sizes and high mutation rates increased the evolution of suicidal altruism (Johnson et al., 2014a). Additionally, Beckmann et al. studied the evolution of quorum sensing in Avida (Beckmann and McKinley, 2009). They determined that digital organisms could evolve to use quorum sensing to suppress replication when population density reached a specific threshold. Furthermore, they established that organisms using quorum sens-

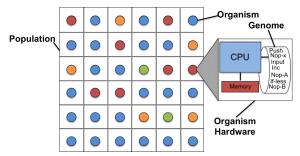


Figure 1: **An Avida population.** Each square represents a cell potentially occupied by an organism. Each circle represents an organism, where color denotes the genotype. Each organism consists of a genome, a virtual CPU for executing instructions, and memory space for storing values.

ing could utilize it within populations up to 400 times larger than those in which the organisms had evolved. While these studies explored the conditions under which suicidal altruism and quorum sensing arise independently, here we investigate whether the conjunction of these traits is beneficial.

Methods

For these experiments, we used the Avida digital evolution platform (Ofria and Wilke, 2004). Within Avida, digital organisms live on a toroidal grid where they compete for space (Figure 1). A digital organism consists of a computer program (its genome) and virtual hardware upon which the program is executed. Included in this virtual hardware are three registers that the organism can use to store and manipulate numbers. Each genome consists of a series of instructions that enable the organism to input numbers from and output numbers to the environment (IO), manipulate numbers, and self-replicate. Each organism occupies a specific cell in the grid. When an organism replicates, it creates two daughter organisms that inherit a potentially-mutated version of their parent's genome. One daughter organism replaces the parent and the other daughter organism is placed into a surrounding cell, potentially killing the current occupant.

Altruism and Quorum Sensing Instructions

For this study, in addition to the standard set of Avida instructions (Ofria and Wilke, 2004), we developed several instructions that enable organisms to evolve to use quorum sensing and altruistically explode (potentially killing competitors).

If evolved, the *quorum-sense* instruction enables the digital organism to sense how many related organisms surround it. For these experiments, we consider two organisms to be *related* if their genomes differ by three or fewer instructions. An organism's neighborhood consists of any organisms in the 5x5 grid surrounding the organism. When an organism executes the *quorum-sense* instruction, the following computation occurs: (1) The proportion of neighboring

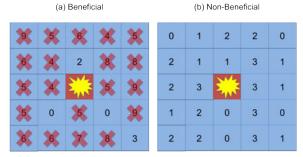


Figure 2: The results of an organism exploding in a beneficial and non-beneficial environment. Each 5x5 grid represents a subsection of the Avida world, where each square represents an organism. Within the grid, the center square represents the focal, exploding organism. The numbers specify the number of genomic differences from the focal organism. Squares containing an X represent organisms that are killed by the explosion because they are unrelated to the focal organism (i.e., they have four or more genomic differences). (a) This grid depicts the effects of a beneficial explosion. Many of the organisms surrounding the exploding organism are unrelated and are killed by the toxin released by the exploding organism. (b) This grid depicts the effects of a non-beneficial explosion. All the organisms in the exploding organism's neighborhood are related, thus the focal organism's explosion did not kill any competitors.

cells filled by related organisms is computed. (2) If this proportion is *less* than a *quorum-threshold* value specified by the organism, then one of the organism's registers is set to TRUE. While the organisms can use this capability in isolation, we envision that they will use it to assess whether environmental conditions are suitable for altruistic explosions.

The smart-explode instruction, if evolved, enables an organism to use quorum information in determining whether or not to explode. In many examples within nature, quorum sensing is used as a trigger for actions (e.g., a sufficient number of bacteria are present to produce an antibiotic). However, for this study, we envision that quorum sensing will be used to inhibit explosions under unsuitable conditions (Figure 2), though the population must still evolve to use the instruction in this way. Specifically, the presence of a quorum indicates that the focal organism is surrounded by related organisms and an explosion would not free up resources. As such, when the *smart-explode* instruction is executed, the following computation occurs (Figure 3): If the register specified by the organism (by default this is the same register that quorum-sense sets) is set to TRUE, the organism explodes with 5% probability. Otherwise, the instruction has no effect. The probability of exploding was 5% to be congruent with previous digital evolution studies (Goings et al., 2004).

Several factors influence whether organisms successfully make use of these instructions to sense their environment

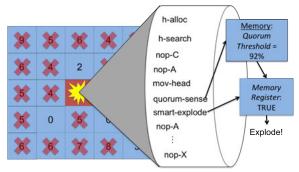


Figure 3: An organism executing the *quorum-sense* and *smart-explode* instructions. A portion of the genome and internal state of the exploding organism (exploding center square) are shown. When the organism executes the *quorum-sense* instruction, a register in the organism's memory is set to TRUE because the percentage of related individuals is less than 92% (the organism's quorum threshold). The organism then executes the *smart-explode* instruction, which checks the same register in memory and, because it is TRUE, explodes with a 5% chance.

and react accordingly. First, organisms are not able to directly sense the presence of competitors, but must infer this information from the proportion of related organisms around them. When an organism detects that only a few kin are in its neighborhood, it cannot be certain that the rest of its neighborhood contains competitors. Some or all of the nonkin cells could be empty spaces instead, rendering an explosion useless. This instruction mimics the quorum-sensing capabilities of natural organisms, who only sense the presence of conspecifics. Additionally, the register used by the smart-explode instruction can also be used by other instructions. An organism could have a suicidal-altruism strategy that does not use quorum sensing. The quorum-sense and smart-explode instructions provide the most accurate information when executed sequentially. This ensures that the organism is using the information gathered by the quorumsense instruction and that that information reflects the current status of the surrounding environment. For example, if an organism executes the quorum-sense instruction early within its lifetime, but waits to execute the smart-explode instruction until substantially later, then the quorum-sense information may no longer be accurate. This could result in an explosion that would not otherwise have occurred.

Experimental Parameters

For each treatment, we configured an environment in which organisms evolved on a 60x60 toroidal grid for 30,000 updates. An update is the time unit used by Avida. One update allows an organism to execute 30 instructions on average and 10 updates is approximately one generation. All experiments had a mutation rate of 0.02 genomic. These are standard parameters employed by Avida experiments (Go-

ings et al., 2004; Goldsby et al., 2012).

We performed two main types of experiments: single-lineage evolution and competition assays. The single-lineage evolution experiments started with one ancestor with a genome of length 100. The ancestor contained only instructions necessary for replication and no-operation (nop) instructions. In the quorum-sensing-altruists treatment of this experiment, the standard set of instructions and the *quorum-sense* and *smart-explode* instructions were available via mutation. In the non-quorum-sensing-altruists treatment (a control), the standard set of instructions and the *explode* instruction were available via mutation. When an organism executes the *explode* instruction, it explodes with a 5% probability.

To assess whether the *quorum-sense* and *smart-explode* instructions were undergoing positive selection, we compared how frequently they were executed to the frequency with which the *nop-Y* instruction was executed. *Nop-Y* is a no-operation instruction that does not affect the operation of the organism and therefore was used as a control.

We performed several competition assays with distinct starting proportions of the different lineages. For these experiments, the organisms were not able to mutate the quorum-sense, smart-explode or explode instructions in or out of their genome, which enabled us to assess the competitive capability of these lineages. The first set of competition assays started with a colony of non-quorum-sensing altruists and a colony of quorum-sensing altruists in equal proportions. The second set of experiments began with 95% non-quorum-sensing altruists and 5% quorum-sensing altruists. This permutation of initial conditions enabled us to test whether quorum-sensing altruists could invade a colony of non-quorum-sensing altruists. Finally, we determined if quorum-sensing altruists could repel non-quorum-sensing altruists by configuring the initial population to contain 5% non-quorum-sensing altruists and 95% quorum-sensing altruists.

Configuration files and data for all experiments are available at https://github.com/anyaejohnson/Communication14.

Results and Discussion

In this study, we first explore whether organisms will evolve to use quorum sensing to inform their decisions about when to behave altruistically. Next, we examine the strategies organisms evolved to determine how many related organisms constitute a quorum. Finally, we determine if organisms that can use quorum sensing to inform their decisions regarding if and when to altruistically commit suicide have a competitive advantage over organisms that do not have access to quorum-sensing information.

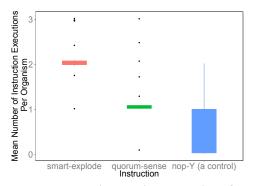


Figure 4: The average instruction executions for smart-explode, quorum-sense and a neutral instruction per organism in the population. The *smart-explode* instruction was used 1.95 more per organism than the neutral instruction (nop-Y) in the last 100 updates. The *quorum-sense* instruction was used 0.99 more per organism than the neutral instruction (nop-Y) in the last 100 updates (95% CI for difference between smart-explode and neutral instruction is 1.92 to 1.97 executions per organism, Wilcox rank sum test p < 0.0001; 95% CI difference *quorum-sense* instruction executions and neutral instruction is 0.96 to 1.00 executions per organism, Wilcox rank sum test p < 0.0001, error bars represent 95% confidence intervals). Both quorum sensing and suicidal altruism are under positive selection in this environment.

Can organisms evolve to use quorum information to decide when to commit suicide?

For our first experiment, we investigated whether quorumsensing and suicidal-altruism behaviors would evolve, rise in frequency, and ultimately stabilize in a population. To assess whether these capabilities were beneficial, we evolved a colony of organisms from an ancestor organism that could only reproduce, but quorum-sensing and suicidal-altruism instructions were available via mutation.

Figure 4 shows the execution rates per organism of the quorum-sensing, suicidal-altruism, and neutral instructions. In agreement with previous work (Goings et al., 2004), suicidal altruism was used an average of 1.95 time more frequently than a neutral instruction per organism by the final time point (median number suicidal-altruism instruction executions per organism = 2.00, neutral instruction executions per organism = 0.04, 95% confidence interval of difference is 1.92 to 1.97 instruction executions, Wilcox rank sum test p < 0.0001). Organisms also evolved to use the quorumsensing instruction an average of 0.99 more frequently than the neutral instruction per organism by the final time point (median number of quorum-sensing executions per organism = 1.03, neutral instruction executions = 0.04, 95% confidence interval of difference is 0.96 to 1.00 more instruction executions per organism, Wilcox rank sum test p < 0.0001). These results demonstrate that both suicidal-altruism and quorum-sensing capabilities were selected for.

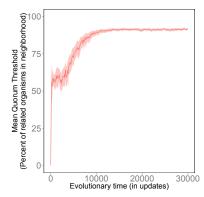


Figure 5: Mean quorum threshold evolved by quorum-sensing altruists. The quorum-sensing altruists evolved to use a *quorum threshold* of approximately 92% related organisms in their neighborhood. Within the context of this study, it is likely that an organism using quorum sensing would explode if its neighborhood contained 8% or more unrelated organisms or empty spaces.

How many related organisms constitute a quorum?

Quorum-sensing altruists are required to set a threshold to determine how many organisms constitute a quorum. This quorum threshold also implicitly sets the number of potential competitors or empty spaces that must be present for an explosion to be considered beneficial. For our quorum-sensing-altruist treatment, we explored how many organisms constituted a quorum by recording the value of the quorum threshold each time an organism exploded (once the exploding mechanism evolved into the population).

Figure 5 shows that the organisms evolved a median quorum threshold that corresponded to approximately 92% of the organisms around the focal organism being related (95% confidence interval is 91.29% to 92.63%). This percentage indicates that organisms using the quorum-sensing instruction with suicidal altruism would only probabilistically explode if one or more unrelated organisms or empty spaces were around the organism. This mechanism prevents the organism from exploding when the organism is completely surrounded by related organisms.

Is quorum information beneficial?

Ideally, an organism should use quorum sensing to assess whether its environmental conditions are such that an explosion would benefit its kin. However, the act of quorum sensing also has a cost: the organism must spend time gathering information about its environment. Within our study, this cost is experienced by the organism as the CPU cycles it devotes to gathering and using quorum-sensing information. To assess whether the costs of using quorum-sensing information outweigh the benefits of additional environmental information, we compared the behavior of quorum-sensing altruists to non-quorum-sensing altruists.

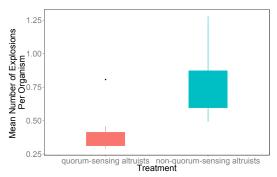


Figure 6: Mean number of explosions per organism for the quorum-sensing altruists and the non-quorum-sensing altruists. At the final time point in the treatments, the quorum-sensing altruists explode 0.35 less per organism on average than the non-quorum-sensing altruists (median quorum-sensing-altruist explosions = 0.35, median non-quorum-sensing-altruist explosions = 0.75, 95% CI for difference is 0.28 to 0.45 explosions per organism, Wilcox rank sum test p < 0.0001).

To investigate how these costs and benefits influence altruistic behavior, we determined the number of explosions per organism that occurred within the quorum-sensing-altruist and non-quorum-sensing-altruist treatments. Figure 6 shows the number of explosions per organism in each treatment. The quorum-sensing altruists explode an average of 0.35 times less frequently per organism than the non-quorum-sensing altruists (median quorum-sensing altruist explosions = 0.35, median non-quorum-sensing-altruist explosions = 0.75, 95% confidence interval of difference is 0.28 to 0.45 fewer explosions per organism, Wilcox rank sum test p < 0.0001). These results may indicate that quorum sensing allows the quorum-sensing altruists to avoid exploding in a detrimental environment or they may indicate that quorum sensing is not adaptive.

To assess whether quorum-sensing altruists are deriving more benefit from each explosion than non-quorumsensing altruists, we compared the mean number of organisms killed per explosion in each treatment. Figure 7 shows that the mean number of organisms killed per explosion by the quorum-sensing altruists and non-quorum-sensing altruists is 2.004 and 1.231, respectively. Thus, on average, the quorum-sensing altruists kill 0.702 more organisms per explosion than by the non-quorum-sensing altruists (95% confidence interval of difference is 0.567 to 0.847 organisms killed per explosion, Wilcox rank sum test p < 0.0001). The population size is also larger in the quorum-sensing-altruists treatment, which could be contributing to the greater number of organisms killed (median population size in the final time point of quorum-sensing-altruists treatment = 3423.5, non-quorum-sensing-altruists treatment = 3371.5). When the mean number of organisms killed is normalized by population size, the quorum-sensing altruists kill an average

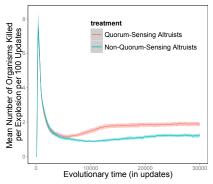


Figure 7: A comparison of the effects of quorum-sensing and non-quorum-sensing-altruist explosions. Shown is the average number of organisms killed per explosion every 100 updates. The quorum-sensing-altruist treatment is more efficient, killing 0.702 more organisms on average than the non-quorum-sensing altruists (mean number of organisms killed per explosion by quorum-sensing altruists = 2.004, mean number of organisms killed per explosion by nonquorum-sensing altruists = 1.231, 95% CI of difference is 0.567 to 0.847, Wilcox rank sum test p < 0.0001). We also calculated the average number of organisms killed per explosion normalized by population size. When normalized by population size, the average number of organisms killed per explosion by quorum-sensing altruists is 0.000197 more organisms than the non-quorum-sensing altruists (mean number of organisms killed per explosion normalized by population size for quorum-sensing altruists = 0.00058, nonquorum-sensing altruists = 0.00037, 95% CI of difference is 0.000161 to 0.000243, Wilcox rank sum test p < 0.0001).

of 0.000197 more organisms per explosion than the non-quorum-sensing altruists (mean number of organisms killed per explosion normalized by population size for quorum-sensing altruists = 0.00058, for non-quorum-sensing altruists = 0.00037, 95% confidence interval of difference is 0.000161 to 0.000243, Wilcox rank sum test p < 0.0001). Therefore, even when larger population size is accounted for, quorum-sensing altruists kill more competing organisms than the non-quorum-sensing altruists on average. These results suggest that the benefits of quorum sensing may outweigh the costs.

Does quorum sensing provide a competitive advantage for suicidal altruists?

We have demonstrated that quorum-sensing altruists kill more organisms per explosion than non-quorum-sensing altruists. However, these results do not yet demonstrate that quorum sensing provides a competitive advantage. While quorum-sensing altruists may kill more organisms per explosion, the costs of quorum sensing may yet outweigh this benefit. Therefore, we performed competition assays in which quorum-sensing altruists compete directly against

non-quorum-sensing altruists. Our competition assays used two lineages of organisms. One lineage was required to use the quorum-sensing-altruist strategy and the other lineage was required to use the non-quorum-sensing-altruist strategy. To focus on the competitive capabilities of these two strategies, we hand wrote both ancestor organisms and prevented the quorum-sensing and suicidal-altruism instructions from being accessible via mutation.

For our first competition assay, we filled half the population with copies of the quorum-sensing-altruist ancestor and the other half with copies of the non-quorum-sensing-altruist ancestor. Figure 8a shows the mean percent of organisms of each lineage over time. Out of 30 replicates, the quorum-sensing lineage fixed 30 times. Therefore, despite the additional costs of quorum sensing, the information gathered through quorum sensing provides a competitive advantage.

There are many situations in nature where a colony of organisms already exists. Therefore, we examined whether or not the quorum-sensing altruists could invade an existing population of non-quorum-sensing altruists. Specifically, we performed competition assays in which the lineage of quorum-sensing altruists started at only 5% of the population and the non-quorum-sensing altruists started at 95% of the population. As seen in Figure 8b, the quorum-sensing altruists were not able to invade a pre-established population of the non-quorum-sensing altruists and went extinct in 30/30 replicates.

Finally, we examined the reverse situation and explored whether quorum-sensing altruists could repel invading non-quorum-sensing altruists. Figure 8c depicts our results. In this case, the quorum-sensing altruists are able to repel the non-quorum-sensing altruists in all 30 replicates. These experiments demonstrate that quorum-sensing altruists have a competitive advantage over non-quorum-sensing altruists when they start at an equal or greater proportion of the population.

Conclusions

We explored whether organisms would evolve to use quorum sensing to inform a suicidal altruism strategy and whether such a strategy would have a competitive advantage. We found that when we enable digital organisms to set their own quorum threshold, they evolved a threshold of 92.1% on average, meaning that organisms evolved to require at least one competitor or empty space to be in their neighborhood in order to explode. Additionally, we found that quorum sensing increased the benefit of suicidal altruism (i.e., the number of competitors killed per explosion significantly increased). Finally, we found that quorum-sensing altruists outcompeted non-quorum-sensing altruists when starting at equal proportions or in the majority. Future work could explore intermediate values to address whether quorumsensing altruists could invade a pre-existing population. In further experiments, we also explored how the accuracy of

the quorum-sensing mechanisms affected their use (Johnson et al., 2014b). In general, quorum-sensing accuracy was correlated with an advantage in competitions against non-quorum-sensing altruists.

Suicidal altruism is a critical aspect of survival strategies in many natural systems, despite its high cost (Chao and Levin, 1981; Shorter and Rueppell, 2011). Communication has often been implicated as the element that might aid the evolution of altruism (West et al., 2007; Turner and Chao, 1999; Strassmann et al., 2011). This study offers support that: (1) Quorum-sensing capabilities can be used to make altruistic acts more effective. (2) Communicating altruists can outcompete altruists that are not able to communicate. In this study we purposefully kept the cost of quorum-sensing low to facilitate evolution. Future work could explore how increasing this cost affects the competitive advantage of quorum-sensing altruists.

Acknowledgments

We thank Charles Ofria, David Bryson, and the Devolab for their insightful contributions to this work. This research was supported by funding from the BEACON Center for the Study of Evolution in Action and the NSF. This material is based in part upon work supported by the National Science Foundation under Cooperative Agreement No. DBI-0939454 and No. OCI-1122620. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

- Bassler, B. L. (2002). Small Talk. Cell, 109(4):421-424.
- Beckmann, B. E. and McKinley, P. K. (2009). Evolving quorum sensing in digital organisms. In *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 97–104. ACM Request Permissions.
- Berngruber, T. W., Lion, S., and Gandon, S. (2013). Evolution of suicide as a defence strategy against pathogens in a spatially structured environment. *Ecology Letters*, 16:446–453.
- Bordereau, C., Robert, A., Van Tuyen, V., and Peppuy, A. (1997). Suicidal defensive behaviour by frontal gland dehiscence in Globitermes sulphureus Haviland soldiers (Isoptera). *Insectes Sociaux*, 44(3):289–297.
- Bowles, S. (2006). Group Competition, Reproductive Leveling, and the Evolution of Human Altruism. *Science*, 314(5805):1569–1572.
- Chandler, J. R., Heilmann, S., Mittler, J. E., and Greenberg, E. P. (2012). Acyl-homoserine lactone-dependent eavesdropping promotes competition in a laboratory co-culture model. *The ISME Journal*, 6(12):2219–2228.
- Chao, L. and Levin, B. R. (1981). Structured habitats and the evolution of anticompetitor toxins in bacteria. *Proceedings of the National Academy of Sciences*, 78(10):6324–6328.

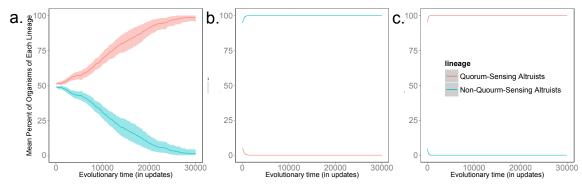


Figure 8: Mean percent of quorum-sensing altruists (red) and non-quorum-sensing altruists (blue) over time. In (a), each lineage started at 50% of the population and then evolved in competition. In (b), the quorum-sensing altruists started at 5% of the population and the non-quorum-sensing altruists started at 95%. In (c), the quorum-sensing altruists started at 95% of the population and the non-quorum-sensing altruists started at 5%. The quorum-sensing altruists achieved fixation by the end of the experiment when starting at equal proportion or the majority proportion. The non-quorum-sensing altruists achieved fixation when they started in the majority.

- Crespi, B. J. (2001). The evolution of social behavior in microorganisms. *Trends in Ecology & Evolution*, 16(4):178–183.
- Davies, D. G. (1998). The Involvement of Cell-to-Cell Signals in the Development of a Bacterial Biofilm. *Science*, 280(5361):295–298.
- Diggle, S. P., Griffin, A. S., Campbell, G. S., and West, S. A. (2007). Cooperation and conflict in quorum-sensing bacterial populations. *Nature*, 450(7168):411–414.
- Dworkin, M. and Kaiser, D. (1985). Cell interactions in myxobacterial growth and development. *Science*, 230(4721):18–24.
- Foster, K. R., Wenseleers, T., and Ratnieks, F. L. W. (2006). Kin selection is the key to altruism. *Trends in Ecology & Evolution*.
- Goings, S., Clune, J., Ofria, C., and Pennock, R. T. (2004). Kin selection: The rise and fall of kin-cheaters. In *Proceedings of* the Ninth International Conference on Artificial Life, pages 303–308.
- Goldsby, H. J., Dornhaus, A., Kerr, B., and Ofria, C. (2012). Task-switching costs promote the evolution of division of labor and shifts in individuality. *Proceedings of the National Academy of Sciences*, 109(34):13686–13691.
- Hamilton, W. D. (1963). The Evolution of Altruistic Behavior. *The American Naturalist*.
- Hamilton, W. D. (1964). The genetical evolution of social behaviour. I. *Journal of theoretical biology*.
- Johnson, A. E., Goings, S., Goldsby, H. J., and Ofria, C. (2014a). Exploring the Evolution of Kin Inclusivity using Digital Organisms. In GECCO '14: Proceedings of the 16th annual conference on Genetic and evolutionary computation. ACM.
- Johnson, A. E., Strauss, E., Pickett, R., Adami, C., Dworkin, I., and Goldsby, H. J. (2014b). More bang for your buck: Quorumsensing capabilities improve the efficacy of suicidal altruism. Technical Report MSU-CSE-14-2, Computer Science and Engineering, Michigan State University, East Lansing, Michigan.

- Kerr, B., Godfrey-Smith, P., and Feldman, M. W. (2004). What is altruism? Trends in Ecology & Evolution, 19(3):135–140.
- Khare, A., Santorelli, L. A., Strassmann, J. E., Queller, D. C., Kuspa, A., and Shaulsky, G. (2009). Cheater-resistance is not futile. *Nature*, 461(7266):980–982.
- Lenski, R. E., Ofria, C., Pennock, R. T., and Adami, C. (2003). The evolutionary origin of complex features. *Nature*, 423:139– 144.
- Miller, M. B. and Bassler, B. L. (2001). Quorum sensing in bacteria. *Annual Reviews in Microbiology*, 55:165–199.
- Ofria, C. and Wilke, C. O. (2004). Avida: A Software Platform for Research in Computational Evolutionary Biology. *Artificial Life*, 10(2):191–229.
- Shorter, J. R. and Rueppell, O. (2011). A review on self-destructive defense behaviors in social insects. *Insectes Sociaux*, 59(1):1–10.
- Strassmann, J. E., Gilbert, O. M., and Queller, D. C. (2011). Kin Discrimination and Cooperation in Microbes. *Annual Review of Microbiology*, 65(1):349–367.
- Traulsen, A. (2006). Evolution of cooperation by multilevel selection. *Proceedings of the National Academy of Sciences*, 103(29):10952–10955.
- Turner, P. E. and Chao, L. (1999). Prisoner's dilemma in an RNA virus. *Nature*, 398(6726):441–443.
- Velicer, G. J., Kroos, L., and Lenski, R. E. (2000). Developmental cheating in the social bacterium Myxococcus xanthus. *Nature*, 404(6778):598–601.
- West, S. A., Griffin, A. S., and Gardner, A. (2007). Social semantics: altruism, cooperation, mutualism, strong reciprocity and group selection. *Journal of Evolutionary Biology*, 20(2):415–432.

Structured Populations with Limited Resources Exhibit Higher Rates of Complex Function Evolution

Arthur W Covert III^{1,2}, Siena McFetridge^{1,2} and Evan DeLord^{1,2}

¹University of Texas at Austin, Center for Computational Biology and Bioinfomatics ²Beacon Center for the Study of Evolution in Action art.covert@austin.utexas.edu

Abstract

The impact of population structure on evolving populations is difficult to study. Populations broken up into groups of organisms and connected by low levels of migration will experience different types of geneflow than normal unstructured populations. Various studies, spanning decades of research, have lead to seemingly contradictory conclusions. Some point to population structure as a means to improve adaptation, others argue that population structure hinders evolution. We investigate how population structure impacts the evolution of complex functions in environments with limited resources. We find that structured populations with limited resources tend to evolve complex functions at a higher rate than unstructured populations, across a broad range of migration rates. This suggests that population structure may have an important impact on evolution, in both sexual and asexual populations, at least at certain migration rates.

Introduction

Few topics in evolutionary biology have been as hotly debated as the impact of population structure (Moore and Tonsor 1994, Phillips 1996, Coyne et al. 1997, Wade and Goodnight 1998, Coyne et al. 2000, Goodnight and Wade 2000, Kryazhimskiy et al. 2012, Covert and Wilke 2014). Sewall Wright first suggested that populations structured into isolated subpopulations, connected with low levels of migration, created a balance between exploration of fitness landscapes and exploitation of fitness peaks (Wright 1932). According to Wright's theory, these structured populations could pass through fitness valleys that large unstructured populations could not.

Evolutionary computation researchers have relied on structured populations to improve the speed of evolutionary search, though little work has been done to determine if the improvement comes from a process similar to the one Wright envisioned, or some other source (Lin et al 1994, Belding 1995, Cantu-Paz 1998, Cantu-Paz 2001, Fernándex et al 2003). Meanwhile, the notion that structured populations may improve adaptation has proved controversial in evolutionary biology (Conye et al 1997, Coyne et al 2000), with many scientist arguing that populations structure may have no effect, or a negative effect on rates of adaptation (Kryazhimskiy 2012). Other studies have suggested that population structure may have a very important impact under

certain conditions (Moore and Tonsor 1994, Wade and Goodnight 1998, Goodnight and Wade 2000, Covert and Whilke 2014).

In structured populations, beneficial mutations take longer to sweep because they must wait for migration to carry them to every subpopulation. The longer a beneficial mutation takes to sweep, the greater the chance that a superior beneficial mutation will be discovered in one of the subpopulations that has not yet been swept yet. This process has been observed in large structured asexual populations and is referred to as leapfrogging (Gerrish and Lenski 1998, Miller et al 2011, Covert and Wilke 2014).

In a recent paper, Covert and Wilke (2014) demonstrated that at certain migration rates, leapfrogging from older genetic backgrounds yielded a dramatic fitness improvement. However, these experiments were asexual and did not address the impact of population structure in sexual populations. In addition, Covert and Wilke's experiments simulated populations with unlimited resources, which was a necessary simplifying assumption for their work, but is ultimately biologically unrealistic.

In this paper, we examine the impact of population structure in environments with limited resources. We run experiments with self-replicating digital organism that may reproduce asexually or sexually. We demonstrate that population structure improves the adaptive ability of populations, even when resources are limited, under both methods of reproduction.

Methods

We used Avida 2.12.2 (Ofria and Wilke 2004), to evolve self-replicating digital organisms, our complete experimental setup is free to download from git hub. Digital organisms are computer programs that exist on a grid. Each grid cell contains a virtual CPU and memory for the organisms to execute their code. During their lifetime each organism must copy every instruction in its genome into a new memory space and then execute a special "divide" command that places the new offspring in the environment. Single point mutations had a 25% probability of occurring on divide. All of our

populations were seeded with a hand-written ancestor 51 instructions long that could do nothing initially but copy itself.

All environments had nine logical functions that were rewarded in terms of increasing complexity and resource availability (Lenski et al 1999, Chow et al 2001, Covert et all 2012). Digital organisms could perform each of the nine logical functions by evolving a series of NAND operations. When a function was successfully performed the organisms' were rewarded only if there was sufficient resource in their current cell. When sufficient resources were present, they would be consumed by the execution of the function and the organism would be allowed to run its genome faster (Table 1). Rewards for each logical function were proportional to the number of NAND operations required to execute the function's most parsimonious configuration. Standard Avida populations have a difficult time evolving the most complex functions: XOR and EQU (Lenski et al. 2003, Covert et al 2012).

Function Name	Logic Operation	Speed Increase
NOT	~A; ~B	x2
NAND	~(A AND B)	x2
AND	A AND B	x4
OR_N	(A OR ~B) (~A OR B)	x4
OR	A OR B	x8
AND_N	(A AND ~B) (~A AND B)	x8
NOR	~A AND ~B	x16
XOR	(A AND ~B) OR (~A AND B)	x16
EQU	(A AND B) OR (~A AND ~B)	x32

Table 1 The standard nine logical functions in the Avida environment and their speed increases. Digital organisms have only the NAND operation in their instruction sets and must construct other logical functions out of NAND operations. The energy bonus for each function is equivalent to 2ⁿ, where n is the minimum number of NAND operations needed to complete it. In our experiments, each function reward was granted only if the organism was able to consume 1 unit of a limited resource.

We ran two experiments to test the effects structured populations with limited resources, one sexual experiment and one asexual experiment. Each experiment had 4 treatments, 3 in which the population was structured into 100 demes each with 100 organisms, and 1 that was a single large deme of 10,000 organisms. Each treatment had 40 replicates, seeded with the default ancestor and a unique random number seed.

Migration between demes occurred when digital organisms were born. Each population ran for 250,000 updates.¹

Each organism had a small probability of being placed in a different deme when it divided from its parent. Migration rates were selected based on previous results (Covert and Wilke 2014) indicating that the optimal migration rate was around 5x10⁻⁵ (one migration every other generation). We ran additional migration rates at one order of magnitude higher, 5x10⁻⁴ (5 migrants every generation) and one order of magnitude lower, 5x10⁻⁶ (one migrant every 20 generations).

In the first experiment, all organisms reproduced asexually, with offspring being placed in an adjacent grid cell. In the second experiment, all organisms reproduced via sexual recombination, similar to the recombination in Misivic et al (2006). In the sexual experiments, organisms copied their genomes and then placed their offspring into a "birth chamber". Each birth chamber was associated with the organism's deme. When an organism was placed in the birth chamber, it waited to be joined by a second offspring organism. When two organisms were present in the birth chamber, their genetic code was divided up into 5 regular segments, each segment had a 50% chance of being exchanged with the corresponding segment in the other genome. Finally the new organisms are placed in the population, adjacent to one of their parents. Each time a sexual organism is placed in the environment they undergo migration the same way asexual organisms do. Sexual organism being placed in the environment there is a small random probability that it will migrate different deme and be placed there.

We measured three main metrics in all experiments: fitness, genetic diversity, and how many complex functions evolved. Fitness was a ratio of total cpu speed up and the time an organism took to make a copy of itself. We recorded fitness of the most abundant genotype at the end of each experiment, this genotype is referred to as the final dominant. Diversity was measured in terms of Shannon information entropy on the number of genotypes in the population. This equation is given below (equ 1), where p_i is the proportion of the ith genotype in the population. Populations that had more genotypes have higher information entropy, populations with fewer genotypes have lower information entropy. Finally, we counted how frequently XOR and EQU, the two most complex functions (Lenski et al 1999, Lenski et al 2003), were performed. Unstructured populations of digital organisms with this size and mutation rate will only rarely evolve all nine logical functions (Covert et al 2013, Covert and Wilke 2014).

$$H(genotypes) = -\sum p_i \log_2 p_i \tag{1}$$

In all treatments resources were limited to an inflow of 10,000 units per update, distributed uniformly across the population so that each cell in the grid received one unit of resource. Each organism could use 1 unit of resource to perform one logical function during its life cycle. If no resource was

¹Updates are a unit of time in Avida. Each update the population executes 30 CPU cycles per living organism. CPU cycles are awarded based on how many logical functions the organism completes successfully.

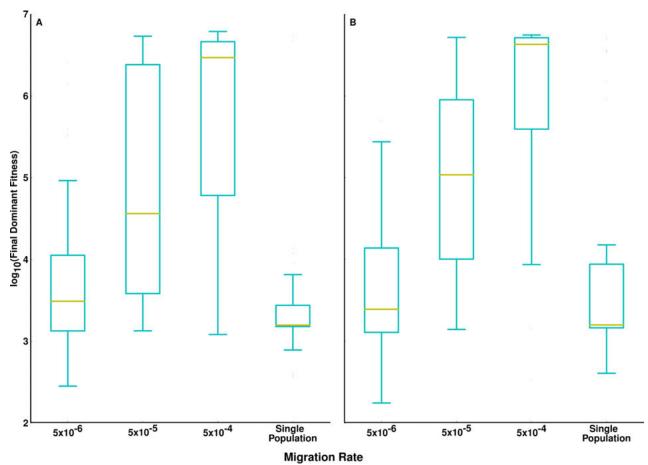


Figure 1 Log₁₀ Final dominant fitness at the end of each experiment for sexual (A) and asexual (B) experiments. All structured populations had higher median fitness than the single population controls. Differences in the $5x10^{-4}$ and $5x10^{-5}$ migration rates were significant, while the $5x10^{-6}$ migration rate was not significantly different from the control. See text for full statistical analysis.

available then no reward was granted for the function. Resources were spatially distributed such that each cell on the grid had its own pool of resource. Past works in Avida, have used globally distributed pools of resources (Chow et al 2006, Walker and Ofria 2012). Allocating resources to individual grid cells meant that organisms in different demes could not utilize the same pool of resources.

Results

Final dominant fitness from all treatments reveals a clear trend, in both sexual and asexual populations (Figure 1). At the highest migration rate, 5×10^{-4} , both sexual and asexual populations evolve higher fitness than in any other treatment (Figure 1), we refer to this migration rate as the optimal migration rate. Both sexual and asexual treatments have significantly higher fitness at the optimal migration rate (p < 0.05 and p < 0.001 respective Mann-Whitney U-test for comparisons with all other treatments).

Fitness remains elevated in the other structured populations, however only the 10^-5 migration rates was significantly

higher than the control populations (both p<0.05 Mann-Whitney U-test). As migration rate decreased, so did fitness, suggesting that genetic drift began to take over in the individual demes as geneflow decreased. This results is constant with pervious results in structured populations (Covert and Wilke 2014), however the range of optimal migration rates is larger here than was perviously observed.

We also measured genetic diversity, as a function of information entropy on the number of genotypes (see methods). Genetic diversity was inversely correlated with fitness in all of the structured populations, both asexual and sexual (Figure 2). Sexual structured populations had significantly higher genetic diversity than asexual populations (all p<0.001, Mann-Whitney U-test). The asexual unstructured population had no significant difference in diversity with the optimal migration rate (p=0.155, Mann-Whitney U-test), but lower migration rates had significantly higher diversity (p<0.01, Mann-Whitney U-test). However, the sexual unstructured population had higher diversity than the sexual structured populations (p<0.05, Mann-Whitney U-test). Despite increased genetic diversity, sexual populations were

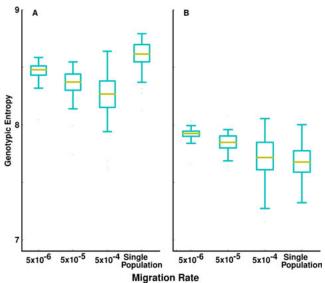


Figure 2 Genotypic entropy of the final population of sexual (A) and asexual (B) experiments. Structural populations had an inverse relationship with with migration rate. All sexual populations have significantly higher fitness than their asexual counter parts. The single population control had significantly higher diversity in the sexual case. The asexual control population had lower diversity or no significant difference in diversity from the asexual structured populations. See text for full statistical analysis.

unable to achieve higher fitness than their asexual counterparts, and the unstructured sexual population had lower fitness than all of structured treatments.

We measured how frequently the two most complex functions, XOR and EQU evolved in each treatment (Figure 3). Structured environments evolved complex function significantly more than the unstructured controls did (all p<0.001 Fisher's Exact test). Within just the structured treatments, asexual treatments at the two highest migration rates had no significant difference in the evolution of complex functions(XOR: p=0.79, EQU: p=1.0). The lowest asexual migration rate tested had fewer complex functions than the other two, but those differences were not significant (p=0.33 and p=0.55). The lowest asexual migration rate still evolved more complex function than the control (both p<0.05). Sexual structured populations followed a similar trend, with the two highest migration rates exhibiting high rates of task evolution and were not significantly different from one another (XOR: p=0.40, EQU: p=0.76). Even the lowest sexual migration rate produced more populations performing complex functions than the unstructured control.

While complex functions evolved at high rates over all migration rates, fitness remained relatively low at lower migration rates. This could indicate that high migration rates bring fit organisms to demes with pools of resources with greater frequency, keeping at least one subpopulation at high fitness at all times. Either way, structured populations had significantly improved evolution.

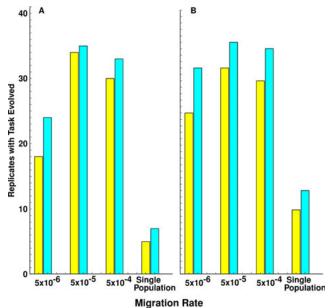


Figure 3 Complex function counts for all sexual (A) and asexual (B) treatments, showing the number of times of XOR (yellow) and EQU (cyan) evolved. All structured populations had significantly improved fitness over their controls. The lowest migration rate, $5x10^{-6}$, had significantly fewer complex functions than the higher migration rates, but was still significantly higher than the single population control in both the sexual and asexual cases. See text for full statistical analysis.

Discussion

Structured populations with limited resources demonstrate improved evolution over their unstructured counter parts. Both sexual and asexual structured populations exhibited improved fitness, genetic diversity and evolved more complex functions. Complex functions evolved frequently over a broad range of migration rates, even though resource limitations depressed overall fitness, particularly at lower migration rates.

Previous results in unlimited resource structured populations have shown an improved performance only at certain optimal migration rates (Covert and Wilke 2014). The optimal migration rate observed here was an order of magnitude higher than that of the previous study, indicating that the rate of resource inflow may determine which migration rate was optimal.

Other migration rates in unlimited environments were not significantly different from the control, or significantly lower than the control at very low migration rates. Structured populations evolved complex functions at a higher rate than the unstructured control, and at a broad range of migration rates. While overall fitness may be depressed due to a lack of resources, complex functions still evolved at a high rate. From this we may conclude that structured populations with limited resources are better able to adapt to their environment than

unstructured populations, or structured populations with unlimited resources.

Our populations with a simple structure and only a single limited resource evolved complex functions at extremely high rates. Evolving different complex functions in the same population is an extremely difficult problem, requiring organisms to maximize their evolutionary potential, while still exploiting already discovered fitness peaks. Evolutionary computation researchers have devised many ways of achieving this maximization through complex ecologies (Goings et al 2012) or through structured populations that impose artificial rules on migration between subpopulations (Hu et al 2005). Our results suggest that a balance between exploration and exploitation may be achieved by mirroring relatively straightforward structures and resource distributions inspired by natural systems.

These results could be altered by a number of factors. Lower resource inflow rates could result in reduced evolution. Here we ensured that every organisms could get rewarded for at least one function execution every update. At lower resource inflow rates selective pressure for complex functions would be reduced because rewards for evolving complex functions would become increasingly rare. At higher resource inflow rates populations would begin to evolve more like populations with infinite resources. Our work indicates that there is a range of optimal inflow rates, where a broad range of migration rates gives a distinct advantage to structured populations over unstructured populations.

Overall, structured populations are better able to adapt to limited resource environments than unstructured populations. This effect is detectable at a large range of migration rates and clearly indicates that resource limitation can dramatically improve evolutionary search in structured populations. This result addresses a long standing controversy in evolutionary biology, indicating that structured populations spread out over a large range may have improved adaptive ability. In addition, it provides further insight into methods that could be used to address difficult problems in computational evolution.

Acknowledgments

We thank Claus Wilke and Charles Ofria for useful discussions, and the Univeristy of Texas at Austin FRI program. This material is based in part upon work supported by the National Science Foundation under Cooperative Agreement No. DBI-0939454. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Belding T.C., 1995 The distributed genetic algorithm revisited. In Eschelman L. Ed., Proceedings of the Sixith International Conference on Genetic Algorithms,114-121, Morgan Kaufmann San Fransico.

- Cantú-Paz, E. 1998 A survey of parallel genetic algorithms. Calculateurs Paralleles, Reseaux et Systems Repartis. 10, 141-171.
- Chow S.S., Wilke C.O., Ofria C., Lenski R.E., and Adami C. (2004). Adaptive radiation from resource competition in digital organisms. Science 305:84-86.
- Covert A.W., Carlson-Stevermer J., Derryberry D.Z., Wilke C.O. 2012. The Role of Deleterious Mutations in the Adaptation to a Novel Environment. In Artificial Life 13, eds Adami C, Bryson D, Ofria C, and Pennock RT (MIT Press) pp 27-31.
- Covert A.W. and Wilke, C.O. (2014) Intermediate Migration Yields Optimal Adaptation in Structured, Asexual Populations. bioRxiv doi:10.1101/003897
- Coyne, J. A., Barton, N. H., Turelli, M., 1997. Perspective: A Critique of Wright's Shifting Balance Theory of Evolution. Evolution. 51: 643-671
- Coyne, J. A., Barton, N. H., Turelli, M., (2000) Is Wright's Shifting Balance Process Important in Evolution? 54:306-317
- Fernández, F., Tomassini, M., & Vanneschi, L. 2003 An empirical study of multipopulation genetic programing. Genetic Programing and Evolvable Machines 4, 21-51.
- Gerrish, P. J. and Lenski, R. E., 1998. The Fate of Competing Beneficial Mutations in an Asexual Population. Genetica, 102/103: 127-144.
- Goodnight, C. J., and M. J. Wade. 2000. The ongoing-synthesis: a reply to Coyne et al. (1999). Evolution 54:317-324.
- Kryazhimskiy, S., Rice, D. P., Desai, M. M., 2012. Population Subdivision and Adaptation in Asexual Populations of Saccharomyces Cerevisiae. Evolution, 66:1931-1941.
- Lenski R.E., Ofria C., Collier T.C., Adami C. 1999. Genome Complexity, Robustness and Genetic Interactions in Digital Organisms. Nature 400:661-664.
- Lenski R.E., Ofria C., Pennock R.T., Adami C. 2003. The Evolutionary Origin of Complex Features. Nature. 423:139-144.
- Lin, S. C., Punch, W. F., & Goodman, E. D. 1994 Coarse-grain parallel gentic algrorithms: categorization and new approach. Proceedings of the Sixith IEEE Symposium on Parallel and Distributed Processing. 28-37, IEEE
- Miller, C. R., Joyce, P., Wichman, H.A., 2011. Mutational Effects and Population Dynamics During Viral Adaptation Challenge Current Models. Genetics, 187:185-202.
- Misevic D., Ofria C. A., Lenski, R. E., 2006 Sexual reproduction reshapes the genetic architecture of digital organisms. Proc. R. Soc. London B 273, 457-464.
- Moore, F. B. G. and Tonsor S. J., 1994. A Simulation of Wright's Shifting Balance Process and the Three Phases. Evolution, 44:69-80.
- Ofria, C. and Wilke, C.O., 2004. Avida: A Software Platform for Research in Computational Evolutionary Biology. J. Artif. Life, 10:191-229.
- Phillips, P. C. (1996) Waiting for a Compensatory Mutation: Phase Zero of the Shifting-Balance Process. Genet. Res., 67:271-283
- Wade, M. J., Goodnight, C. J., Perspectives: The Theories of Fisher and Wright in the Context of Metapopulations: When Nature Does Many Small Experiments. Evolution, 52: 1537-1553

- Walker B and Ofria C (2012) Evolutionary Potential is Maximized at Intermediate Diversity Levels, Proceedings of the 13th International Conference for Artificial Life, East Lansing, MI.
- Wright, S., 1932. The Roles of Mutation, Inbreeding, Crossbreeding and Selection in Evolution. Sixth International Congress on Genetics, Ithica, New York, 356-366.

Robot and Agent Behavior

RoboGen: Robot Generation through Artificial Evolution

Joshua E. Auerbach, Deniz Aydin, Andrea Maesani, Przemyslaw M. Kornatowski, Titus Cieslewski, Grégoire Heitz, Pradeep R. Fernando, Ilya Loshchilov, Ludovic Daler and Dario Floreano

Laboratory of Intelligent Systems Ecole Polytéchnique Fédérale de Lausanne joshua.auerbach@epfl.ch

Extended Abstract

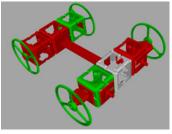
Science instructors from a wide range of disciplines agree that hands-on laboratory components of courses are pedagogically necessary (Freedman, 1997). However, certain shortcomings of current laboratory exercises have been pointed out by several authors (Mataric, 2004; Hofstein and Lunetta, 2004). The overarching theme of these analyses is that hands-on components of courses tend to be formulaic, closed-ended, and at times outdated. To address these issues, we envision a novel platform that is not only a didactic tool but is also an experimental testbed for users to play with different ideas in evolutionary robotics (Nolfi and Floreano, 2000), neural networks, physical simulation, 3D printing, mechanical assembly, and embedded processing.

Here, we introduce RoboGenTM: an open-source software and hardware platform designed for the joint evolution of robot morphologies and controllers a la Sims (1994); Lipson and Pollack (2000); Bongard and Pfeifer (2003). Robo-Gen has been designed specifically to allow evolved robots to be easily manufactured via widely available desktop 3D-printers¹, and the use of simple, open-source, low-cost, off-the-shelf electronic components. RoboGen features an evolution engine complete with a physics simulator, as well as utilities both for generating design files of body components for 3D printing, and for compiling neural-network controllers to run on an Arduino microcontroller board².

In this paper, we describe the RoboGen platform, and provide some metrics to assess the success of using it as the hands-on component of a masters-level bio-inspired artificial intelligence course.

Software Suite

The RoboGen software suite is comprised of two main components: an evolution engine that generates and reproduces robots, and a simulator that renders the evolutionary environment and assesses the fitness of the evolved solutions. Users may go from serial fitness evaluations (using a single



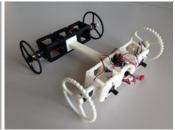


Figure 1: Sample robot evolved with RoboGen: simulation (left) and reality (right).

simulator) to massive parallelism distributed across a network depending on their computational resources.

Robot bodies Robots evolved with RoboGen (see Fig. 1) are composed of predefined and parameterized modules, and are represented as genetic programming trees (Koza, 1992). The modular building blocks that make up the body representations include passive and active structural elements as well as sensing components. A full list of components, and their detailed specifications may be found on the RoboGen website http://www.robogen.org.

Robot brains The "brains" of the RoboGen robots are fully-connected, recurrent artificial neural networks. The robots can sense their environment through touch sensors, light sensors and a six degree-of-freedom inertial measurement unit (IMU). The number of sensors and actuators used in robots increase the complexity of the simulations, but their use may be necessary to evolve robots that are truly adapted to diverse tasks and environments.

In the classroom environment, we provide several scenarios for the students to explore the utility of various parameters and components of the software suite. Specifically, we aim to promote an understanding of how the tasks and environments affect the evolved morphologies (Auerbach and Bongard, 2014), and how allowed simulation complexity changes the adaptedness of the robots generated through the evolutionary process.

¹Such as the MakerBot Replicator 2x: http://store.makerbot.com/replicator2x

²http://www.arduino.cc

The complexity of simulations achievable with this software is entirely dependent on the user: beginners can familiarize themselves with new concepts in a more controlled way by evolving only the neural network controllers, whereas advanced users can work on the evolution engine to customize the evolutionary algorithm or simulator or even introduce new morphological building blocks. This openendedness is a major advantage of the platform and addresses the current concerns regarding science laboratory education (Mataric, 2004). Additionally, our software is also the first educational platform that provides the users with the ability to manufacture their own evolved robots. By allowing users to get completely immersed in the artificial evolution process, we hope to encourage users to think about real-life applicability of their simulations. Finally, we aim to foster collaborations among groups of students with different expertise by having them design evolutionary scenarios, carry out experiments, and test their evolved designs in hardware.

Teaching Assessment

Discerning whether RoboGen is indeed an effective tool for teaching evolutionary robotics requires an analytical approach. For this reason, we devised a measuring tool to get a sense of how well the students in our class meet the desired learning outcomes. In our teaching assessment we focus on a set of measurable learning outcomes that were defined based on the "Content, Skills and Values (CSV)" classification of learning outcomes (Carleton University, 2014). A brief questionnaire was prepared to be administered twice during the course: once after the first in-depth introduction to the RoboGen project (but before the students begin working on the project), and once at the end of the project. The purpose of this scheduling is to determine the improvement in the technical skills targeted in this course.

The evaluation questions fall under one of the aforementioned CSV categories, and are answered on a Likert scale from 1 to 5, with 5 being the strongest positive response (Likert, 1932). The first questionnaire saw 67% participation (53 students).

The psychosocial and environmental factors that influence learning will be measured through a separate questionnaire administered at the end of the course, determining both the students' perception of the classroom environment for learning and their "ideal" environment. This questionnaire is an adapted version of the Science Laboratory Environment Inventory tailored to suit the teaching environment of the course given at EPFL (Fraser et al., 1995).

Conclusions

Overall, we envision RoboGen as not only an effective platform for evolving the morphologies and controllers of manufacturable robots, but also a valuable educational tool. It is a system that should be attractive to researchers, hobbyists, educators and students alike. Going forward we hope to develop a worldwide community around RoboGen. Users will be able to discuss their experiences, share ideas, and contribute to the growth of the project by introducing new morphological building blocks, evolutionary scenarios, and educational exercises.

Acknowledgements

We acknowledge the contributions of the students of MICRO-551 at EPFL, and the members of LIS.

The research leading to these results has received funding partially from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 308943.

References

- Auerbach, J. E. and Bongard, J. C. (2014). Environmental influence on the evolution of morphological complexity in machines. *PLoS computational biology*, 10(1):e1003399.
- Bongard, J. C. and Pfeifer, R. (2003). Evolving complete agents using artificial ontogeny. In *Morpho-functional Machines: The New Species (Designing Embodied Intelligence*, pages 237–258. Springer-Verlag.
- Carleton University (2014). Writing learning outcomes. http://carleton.ca/edc/wp-content/uploads/Writing-Learning-Outcomes.pdf.
- Fraser, B. J., Giddings, G. J., and McRobbie, C. J. (1995). Evolution and validation of a personal form of an instrument for assessing science laboratory classroom environments. *Journal of Research in Science Teaching*, 32(4):399–422.
- Freedman, M. P. (1997). Relationship among laboratory instruction, attitude toward science, and achievement in science knowledge. *Journal of Research in Science Teaching*, 34(4):343–357.
- Hofstein, A. and Lunetta, V. N. (2004). The laboratory in science education: Foundations for the twenty-first century. *Science Education*, 88(1):28–54.
- Koza, J. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Boston, MA.
- Likert, R. (1932). A technique for the measurement of attitudes. *Archives of Psychology*, 22 140:55.
- Lipson, H. and Pollack, J. (2000). Automatic design and manufacture of robotic lifeforms. *Nature*, 406:974–978.
- Mataric, M. J. (Spring 2004). Robotics education for all ages. In Proceedings, AAAI Spring Symposium on Accessible, Handson AI and Robotics Education.
- Nolfi, S. and Floreano, D. (2000). Evolutionary Robotics: The Biology, Intelligence, and Technology. MIT Press, Cambridge, MA, USA.
- Sims, K. (1994). Evolving virtual creatures. In SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques, pages 15–22, New York, NY, USA. ACM.

Collective Design of Robot Locomotion

Mark Wagy and Josh Bongard

University of Vermont, Burlington, VT 05401 mwagy@uvm.edu

Abstract

It has been shown that the collective action of non-experts can compete favorably with an individual expert or an optimization method on a given problem. However, the best method for organizing collective problem solving is still an open question. Using the domain of robotics, we examine whether cooperative search for design strategies is superior to individual search. We use a web-based robot simulation to determine whether groups of human users can leverage design intuition from others to focus search on relevant parts of a complex design space. We show that individuals that work cooperatively with the aid of a simple optimization algorithm are better able to improve the design of robot locomotion than if they were to work individually with the aid of the optimization algorithm. This result suggests that groups of designers may more effectively work in tandem with optimization algorithms than individuals working in isolation.

Introduction

There is a long history of humans interacting with computer systems to design software agents. In one of the most famous examples of interactive design of software agents, human users interacted with Dawkins' *Blind Watchmaker* to generate "biomorphs": increasingly lifelike beings that resulted from a process of interactive selection (Dawkins (1996)). However, there is little basis for groups of humans interacting to design software agents collaboratively.

We know that positive feedback within a collective can spread through the group and result in wider perceptual abilities than that of the individual (Couzin (2007)). These types of positive feedback or autocatalytic phenomena may result in behavior that outperforms what any single agent in a population could achieve. Anecdotally, we know that teamwork in an organization can result in creative approaches to problem solving and that the intelligence of a group cannot be predicted by the average intelligence of its individual members (Woolley et al. (2010)). So might a group working with an optimization algorithm collectively arrive at better performing designs than an individual would on their own? With this question in mind, we use a web-based platform for robot and behavior optimization to determine whether groups of human users can leverage

design intuition from others to focus search on promising regions of a complex design space.

In order for users to be able to easily communicate designs, we have adoped a means of communicating through a language of graphical symbols, which in some sense acts as a "visual programming language". Visual programming structures can result in better "closeness of mapping" than textual representations (Green and Petre (1996)). That is, visual languages map programming abstractions more directly to the domain that they are modeling. We exploited the intrinsic pattern recognition capabilities of humans in the experiments reported here by developing a visual language for robot configuration. Users 'wire' subsets of a robot's joints together using colored lines, which constrains those subsets to move in phase with one another. An underlying evolutionary optimization algorithm then optimizes the movements of the joints to produce locomotion. Subsequent visitors to the site can then ascertain promising areas of the search space by visually inspecting the wiring patterns created by others and contribute their own computer's time to those regions, or create new designs of their own.

Three main literatures informed the design of our experiment: interactive evolutionary algorithms, visual design languages, and collaborative problem solving.

Interactive Evolutionary Algorithms

A group of non-experts, even children, have been found capable of training robots to achieve complex movement. Lund et al. (1998) employed child subjects in a programming-free, interactive evolutionary robotics experiment to train artificial neural networks to produce interesting robot behaviors such as obstacle avoidance and line and wall following. Chernova et al. (2011) used multiplayer online games featuring simulated robot agents to train a Case-Based Reasoning system. The robots then interacted with humans in a physical environment similar to the web-based simulation in which they were trained. However, the focus of most crowdsourcing robotics studies has been on using individual human users to train robots.

Similarly, in studies involving interactive evolutionary robotics, problem solving primarily involved a single individual dedicated to each training instance (Lund and Miglino (1998), Lund et al. (1998), Dozier (2001)). There was a one-to-one relationship between the human interacting with a robot control algorithm, rather than a group of individuals collectively working toward developing better robot control. Typically the individual directly evaluated fitness of computer-generated designs. Takagi (1998) gives a broad overview of how interactive evolutionary computation has been used for engineering and creative domains, from the design of hearing aids to evaluation of 3D CG images. The designs tended to be evaluated by a single human interacting individually with an evolutionary computing process. Celis et al. (2013) demonstrated the use of interactive evolutionary algorithms in robot control to overcome local optima through demonstration. Also, Bongard and Hornby (2013) used human users to train a user model to avoid local optima in an interactive evolutionary algorithm for the control of robot movement. That work built on the user modeling approach to interactive evolutionary robotics pioneered in (Akrour et al. (2011)). Recent work on exploring the possibility of tapping into user creativity using interactive evolutionary algorithms can be seen in Kowaliw et al. (2012) and García-Valdez et al. (2013).

Visual Design Languages

A substantial challenge in collaborative design is effectively communicating complex concepts between members of a group. In order to allow members of a group to build upon prior knowledge and thus improve past designs, key concepts in the problem domain must be communicated through some form of *working memory*. Larkin (1989) suggested that the use of a diagram is a means for storing problem states in a working memory. With this idea in mind, we developed a simple visual language of symbols for robot joint configuration. Users working together use these symbols as a blackboard system (Nii (1986)) to "exchange" ideas and improve the robot design.

Simplification of robot control by defining domain-specific languages is common in the robotics literature. For example, Peterson et al. (1999) define a declarative language for controlling robot behavior. Their framework is also focused on defining robust robot control rather than performing a single task such as fast locomotion. Cox and Smedley (1998), Cox et al. (1998) and Pfeiffer Jr (1998) all successfully employed visual programming languages for robot control. However, in the present work we are using symbols as a means of indirectly constraining the evolution of high-level robot behavior, rather than programming specific responses of the robot to its environment. It is not

the intent of our study to be able to train a robot to exhibit complex behavior with our symbolic cues. Rather, the symbols that represent different classes of robot movement act as a means by which to communicate possible unexplored parts of the evolutionary search space to others. Though in both cases, the importance of the direct mapping of visual indicators to the movement of robots is noted as an important benefit of visual over textual representations of a complex problem such as robot control.

Collaborative Problem Solving

Much of crowdsourcing work is focused on taking a large problem and dividing it into independent "microtasks" to be implemented by independent web-based workers. However, one of the most visible platforms for crowdsourcing is a collaborative one. The crowdsourced online encyclopedia Wikipedia 1 is one of the best examples of the success of collaborative human computing. Collaborative, networked design tasks have also shown promise in past literature. Kan et al. (2001) deployed a virtual reality framework in which the users of the system directly work with each other on product design. This contrasts with the present work in which individuals in the experimental group only indirectly collaborate with each other through the sharing of past designs asynchronously, but without the intent to directly work with each other. Users implicitly work towards a common goal, without directly working together (Doan et al. (2011)), similar to the approach described in (Von Ahn and Dabbish (2004)). Collaborative crowdsourcing of design tasks with an evolutionary computing backbone has also been explored in past work. Yu and Nickerson (2011) crowdsourced selection and recombination in an evolutionary process of designing alarm clocks. shared drawings of alarm clocks among Mechanical Turk workers to iteratively vet more creative and practical designs through a process of drawing combination. They show a significant improvement in the practicality and creativity of designs at the third generation versus the initial designs. The Picbreeder system developed by Secretan et al. (2011) employed a web-based collaborative system to create realistic images using the NEAT algorithm (Stanley and Miikkulainen (2004)). In this paper we investigate whether such collaborative dynamics can be brought to bear on robot design.

Methodology

We deployed a web-based platform (see Figure 1 for a screenshot) in which users "design" different robots, and then an evolutionary optimization algorithm evolves behaviors for them. Users were partitioned into control and

¹www.wikipedia.org

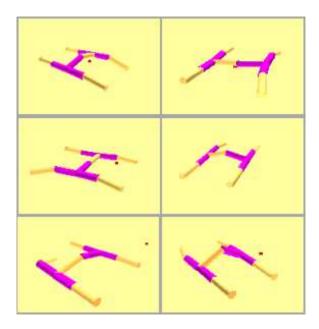


Figure 2: Screenshots the of robot.

experimental groups: those in the control group could not see robots designed by others, while those in the experimental group could. Each user's objective was to create robots that were most amenable to behavior optimization by the evolutionary algorithm. The specific behavior we investigated was legged locomotion.

Robot Form

The robot's form was fixed but users could constrain different subsets of joints to move in phase with one another. This process in essence reduces the dimensionality of the search space for the evolutionary algorithm, as it only has to optimize movements for each joint subset, rather than all joints independently. We henceforth refer to each such user-generated set of joint constraints as a robot design.

The robot's form consisted of ten cylindrical segments, connected with nine hinge joints. The robot had four legs, each with both a "shoulder" joint and an "elbow" joint. Each of the four legs were connected to a "spine" (see Figure 2 for screenshots of robot). The hinge joints could sweep through a range of $[-45^{\circ}, 45^{\circ}]$. The "shoulder" joints rotated the legs through the transverse plane. The "elbow" and "spinal" joints rotated through the sagittal plane.

The robot did not contain any sensors: the evolutionary algorithm produced open-loop controllers for the robots. The robot joints were actuated using a displacement-control sinusoidal signal. This sinusoidal actuation resulted in several possible gaits, depending on the phase value of the

sinusoid that drove each joint. The evolutionary algorithm described below could alter the relative phase between joints, but not the frequency or amplitude. The frequency was set such that the robot typically exhibited 40 rotations of each joint per evaluation. The amplitude allowed each joint to reach just about to its maximum and minimum range. However, this range could be constrained by the momentum of the robot and its friction with the ground plane.

User Interaction

Using a graphical interface representing a top-view of the robot form, users drew lines that linked hinge joints at the robot legs and spine into a group. Each grouping of joints was assigned a distinct color, based on the user-drawn lines that visually linked the nodes representing leg or spine joints (e.g. see Design Panel in lower right corner of Figure 1). When users drew links between joints, they were forcing those joints to have the same phase offset of the sinusoidal control as all others in the joint group - i.e. the phase value in the sinusoidal displacement control signal was the same for all joints in a grouping. This visually-oriented, symbolic display of joint connections allowed for easy communication of intuition about movement of the robot. For example, a user might notice that connecting diagonal joints results in trotting. They may realize that connecting the joints of the front legs together, and connecting the joints of the back legs together produces a two-beat version of cantering. The lines connecting joint group configurations acted as a symbol language that was designed to support the inherent pattern recognition abilities of the humans interacting with the underlying machine learning algorithm.

The robot movement was simulated with the *JBullet*² physics simulation engine, a Java port of the *Bullet Physics Library*³. The user could design their joint grouping in an interactive drawing panel and then press the *Run* button to see a "heads up" or graphical display of robot movement simulated in the *simulation panel*.

The time allowed for each simulation was fixed at 20 seconds. Since there was some level of non-determinism in the simulation, we added *background runs* to increase the sample size for each of the grouping/phase configurations.

Each time a user clicked the "GO" button to launch a "heads up" run (an actual visual simulation of the robot movement), four other background runs with the same joint grouping design and the same associated phase offset values were run, which the user did not see. Fitness was set to the mean distance produced by these five runs. More background runs would have been preferable, but we were

²http://jbullet.advel.cz

³http://bulletphysics.org



Figure 1: Screenshot of web-based design tool. A description of how to use the tool is on the top with an arrow indicating that the ordering of designs is from best to worst. The red arrow below the designs indicates in real-time how the current robot compares to the historical designs displayed. The *GO* button runs a robot simulation and the *reset* button loads a random design. The lower left panel is where the visualization of the robot movement is displayed.

constrained by the need to maintain usability of the tool. Running multiple physics simulation processes at once required enough computing resources that more than four background runs might have degraded the user experience.

Collaborative and Partitioned Groups

When a user opened the application, they were randomly assigned to either a control or experimental group. In the experimental group, users were allowed to see a subset of at most ten past designs created by other users in the experimental group. The ten designs were chosen randomly. If they returned to the site, they would see a different set of ten random designs. Users in the control group were limited to seeing only their own past designs in the history panel. If they ran more than ten designs, ten of their past designs were randomly shown. Users assigned to the control group would remain in the control group even if they left the site and then returned later, and similarly for the experimental group. Since users were anonymous, this group assignment was enforced using their IP address. As such, if a user

were to use the platform from two different computers they could have been exposed to another group. The designs in the history panel were ranked from "worst" (left) to "best" (right) (by best average distance that was achieved for that particular design). The average distance moved by the robot with a given joint grouping was reported under that design. The number of times that particular design had been evaluated was reported above it. Users could use the designs in the history panel as inspiration for how best to improve on the designs and could contribute more runs to a particular design if they wished. They also had the option of generating a random design by clicking the Reset button and altering it into a new design of their making.

Evolutionary Algorithm

The initial values used for each joint group's phase-offset were randomly selected from a uniform distribution between zero and 2π radians. If a given design for a joint grouping was reused, a mutated version of the best past phase values was used in this new simulation. Mutation

was implemented by picking a new value for each grouping with a probability of 0.1. Thus, the optimization used a hill-climber algorithm in which fitness (mean distance) was the basis for drawing from the population of eligible individuals (those that matched a given joint grouping). This best individual was then mutated. The distance that the robot was able to achieve with a given joint grouping and phase offset combination was interpreted as the fitness for that set of phase offsets. When multiple designs were created, the system became a parallel hill climber. Each design corresponds to an individual hill climber, and when a user creates a design that was previously created by another user (or re-created a design she has already created herself), another iteration of that hill climber was executed. If a user created a new design, a new hill climber with an initially random set of phase offsets was assigned to that design. The search space was thus a combination of both the combinatorial design space of joint group configurations and the real-valued space of phase-angle values to assign to each of those joint groupings.

User Incentivization

In order to incentivize users to participate in this experiment - and maintain their continued interest in using it - a sliding pointer under the history panel was incorporated into the user interface. As the robot moved, its real-time distance was indicated by the sliding pointer relative to the visible historical designs in the history panel. This allowed the user to visualize how well their design performed relative to a subset of random past designs, either created by themselves (the control group) or their own and others' designs (the experimental group). A secondary function of the performance slider was to gamify (Deterding et al. (2011)) the platform. We hoped that the user would be motivated to either create new designs that outperformed past designs or provide additional computation to previous designs to improve on them. In short, the user was imbued with the sense that they were in direct competition with a subset of historical designs.

Results

A total of 57 anonymous volunteer users participated in the study via the web. The platform was advertised on online message board systems to attract participants. Participants were not financially rewarded for their participation. Of the users that volunteered, 31 were assigned to the control group and 26 were assigned to the experimental group. The control group ran the simulation a total of 529 times and the experimental group ran the simulation a total of 581 times (see Table 1).

A random sampling of designs created by users in

	Control	Experimental
Number of users	31	26
Number of runs	567	581
Number of designs	76	78

Table 1: User statistics

both the control and experimental groups can be seen in Figure 5. We aggregated the maximum distance values each user was able to achieve. These maximum values were collected per group: the control (n=31) and experimental (n=26). The distributions of the control and experimental group were tested for rejection of the null hypothesis. The null hypothesis in this case was that the events come from the same distribution - that there is no difference between users who worked collectively versus those that worked individually. Using the t-test to reject the null hypothesis under the assumption of normality, we found significant cause to reject the null hypothesis at the 0.05 alpha level (p-value of 0.0404). Testing for the rejection of the null hypothesis without the assumption of normality also gave significant cause to reject the null hypothesis at the $\alpha = 0.05$ significance level, using the non-parametric Wilcoxon Rank Sum test (p-value of 0.0415).

Boxplots of the two distributions can be seen in Figure 3.

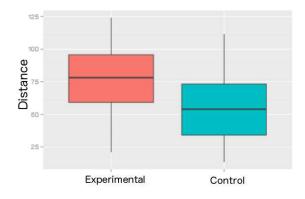


Figure 3: Distance distribution and p-values for significance between experimental and control groups. Significance tests: t-test p-value=0.0406, Wilcoxon Rank Sum p-value=0.0418.

Discussion

We have shown that the control and experimental groups are significantly different at the α level of 0.05. As such, we can say with some degree of confidence that the control and experimental groups are not drawn from the same distribution. That is, the maximum distances achieved by robots designed and optimized by those working collectively was

significantly different than the maximum distances of robots designed and optimized by those working individually. Furthermore, the largest distances achieved by the group working collectively were higher than those achieved by the group of individuals working alone. The experiment thus shows that working in a cooperative setting improved the overall search for better robot designs under these experimental settings.

These results could be a consequence of the fact that the experimental group was exposed to more total designs as a whole. When users of the experimental group first opened the platform, they were exposed to designs from past users (with the exception of the first user). However, exposure to past designs by other users does not necessarily imply that the experimental group has an advantage. The hill climber assigned to a given design may have been able to perform more iterations in the experimental group compared to the hill climber assigned to the same design in the control group, because multiple users in the experimental group may have contributed computational effort to that hill climber. However, this would only advantage the experimental group if this extra effort was directed toward an eventually successful design. Groupthink may have led users in the experimental group to contribute effort to one initially promising design, yet caused them to neglect a design that was less successful at the outset but may have yielded a superior controller in the long run.

It is also possible that the designs to which the experimental group were exposed informed their design decisions, which led to improved designs. Isolating the cause for improved designs in the experimental group will be pursued in future work.

Our experiment involved two embedded search spaces: 1) the combinatorial assignment of joints to phase-locked groups, and 2) the assigning of phase offsets to each group. The users directly searched the first, combinatorial space and the optimization algorithm searched the second, real-valued space. The interaction between the two search spaces may contributed to a stronger separation between the control and experimental groups. Additionally, the simulation of robot movement was not entirely deterministic; although there was a clear separation of mean distances achieved by different configurations. A plot of distributions of one hundred runs for several random configurations can be seen in 4. There is a clear separation of distributions between "good" (large distance traveled) versus "bad" (small distance traveled) configurations as to inform the user's intuition. And this non-determinism and large search space was present in both the control and experimental population so each group operated close to the same conditions.

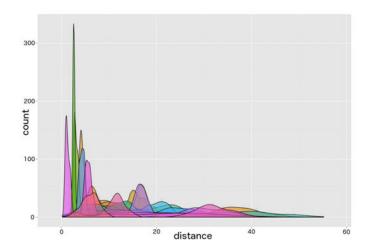


Figure 4: Distance distributions of robot movement for several joint group configurations. Each configuration was run 100 times.

As is common in web-based usage, a small minority of designs attract the majority of computation (Wilkinson (2008)). We can see in Figures 6 and 7 that more users focused their attempts on those designs that were able to move farther. That is, more users ran designs that were able to achieve higher distances in the experimental group; and more design evaluations were dedicated to higher fitness designs. Instead of focusing on designs that exhibited comical or low fitness behavior, users elected to add computing effort to promising, high fitness designs. Although there is not enough data to make definite conclusions about the possibility of "leap-frogging" (the tendency for users to draw inspiration from designs by other users to focus their efforts on promising designs) taking place in the experimental group, the data seem to suggest the possibility of positive feedback contributing to better performance when users worked collectively.

Using human subjects in evaluating fitness and developing new designs presents a number of challenges. When developing the platform, some effort was devoted to gamifying the system. The distance indicator was used to inspire the user to try and "beat" previous designs. Even though written instructions and the distance indicator were used to motivate the user to attempt to evaluate designs based on their ability to cause the virtual robot to move as far as possible, it is possible that users were either unmotivated by the preconceived goal of the simulation or had contrarian intentions to design robots that move as little or as comically as possible.

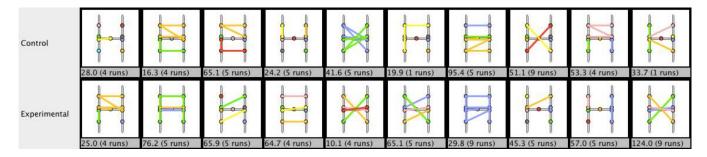


Figure 5: A random sample of designs created by users from the control and experimental groups, shown with best distance achieved and number of times a user evaluated that design (a "run").

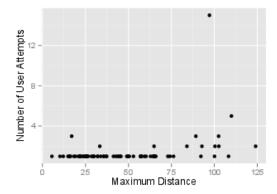


Figure 6: Each point corresponds to a design in the experimental group. The plot shows the number of users that attempted a unique design vs the best distance that that design achieved. Note that the design that received the most attention at 15 attempts was the "default" design that appeared when the application was launched in the user's web browser.

Conclusion and Future Work

We showed that in a hierarchical search space of robot designs in which human users searched the combinatorial level of the space and an optimization algorithm searched the real-valued level, a group that collectively searched the space was able to outperform a group of individuals that worked alone. The distributions of best robot distance achieved per person in each group were significantly different at an α level of 0.05 under both normal and non-parametric assumptions.

Using our framework, the user was shown the best distance that a given robot configuration moved as well as the number of runs contributed to each design. Novelty of a design might also be a useful message to communicate. Users might be incentivized to explore unknown regions of the search space if they were told that they came up with a design that no one else has tried. Additionally, the

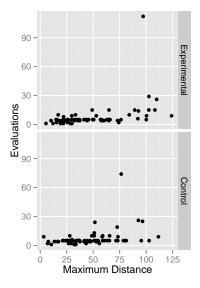


Figure 7: Number of evaluations for each design vs the maximum distance that design achieved. As is the case in Figure 6, the design that received the most evaluations was the "default" design that appeared when the application was launched in the user's web browser.

amount of symmetry in a given design or an indication of the spread of distances realized, such as standard deviation could be communicated. Social networks may be another means of incentivizing search: indicating which designs the user's friends have already explored could be built into the framework.

Another aspect of this work was the use of a symbolic language to easily communicate simple concepts of robot movement between human users. The language was so specific that it would only work with one form of robot. Future work on generalizing this language to quickly communicate complex ideas between collaborators may warrant further investigation.

Acknowledgments

This work was supported by the National Science Foundation (NSF) under project DGE-1144388. This work was also supported by the NSF under grant PECASE-0953837, and by the Defense Advanced Research Projects Agency (DARPA) under grants W911NF-11-1-0076 and FA8650-11-1-7155.

References

- Akrour, R., Schoenauer, M., and Sebag, M. (2011). Preference-based policy learning. In *Machine Learning and Knowledge Discovery in Databases*, pages 12–27. Springer.
- Bongard, J. C. and Hornby, G. S. (2013). Combining fitness-based search and user modeling in evolutionary robotics. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*, pages 159–166. ACM.
- Celis, S., Hornby, G. S., and Bongard, J. (2013). Avoiding local optima with user demonstrations and low-level control. In *Evolutionary Computation (CEC)*, 2013 IEEE Congress on, pages 3403–3410. IEEE.
- Chernova, S., DePalma, N., Morant, E., and Breazeal, C. (2011). Crowdsourcing human-robot interaction: Application from virtual to physical worlds. In *RO-MAN*, 2011 IEEE, pages 21–26. IEEE.
- Couzin, I. (2007). Collective minds. Nature, 445(7129):715-715.
- Cox, P. T., Risley, C. C., and Smedley, T. J. (1998). Toward concrete representation in visual languages for robot control. *Journal of Visual Languages & Computing*, 9(2):211–239.
- Cox, P. T. and Smedley, T. J. (1998). Visual programming for robot control. In *Visual Languages*, 1998. Proceedings. 1998 IEEE Symposium on, pages 217–224. IEEE.
- Dawkins, R. (1996). The blind watchmaker: Why the evidence of evolution reveals a universe without design. WW Norton & Company.
- Deterding, S., Sicart, M., Nacke, L., O'Hara, K., and Dixon, D. (2011). Gamification. using game-design elements in non-gaming contexts. In *PART 2——Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems*, pages 2425–2428. ACM.
- Doan, A., Ramakrishnan, R., and Halevy, A. Y. (2011). Crowd-sourcing systems on the world-wide web. *Communications of the ACM*, 54(4):86–96.
- Dozier, G. (2001). Evolving robot behavior via interactive evolutionary computation: From real-world to simulation. In *Proceedings of the 2001 ACM symposium on Applied computing*, pages 340–344. ACM.
- García-Valdez, M., Trujillo, L., de Vega, F. F., Guervós, J. J. M., and Olague, G. (2013). Evospace-interactive: a framework to develop distributed collaborative-interactive evolutionary algorithms for artistic design. In *Evolutionary and Biologi*cally Inspired Music, Sound, Art and Design, pages 121–132. Springer.

- Green, T. R. G. and Petre, M. (1996). Usability analysis of visual programming environments: a cognitive dimensions framework. *Journal of Visual Languages & Computing*, 7(2):131– 174
- Kan, H., Duffy, V. G., and Su, C.-J. (2001). An internet virtual reality collaborative environment for effective product design. Computers in Industry, 45(2):197–213.
- Kowaliw, T., Dorin, A., and McCormack, J. (2012). Promoting creative design in interactive evolutionary computation. *IEEE transactions on evolutionary computation*, 16(4):523.
- Larkin, J. H. (1989). Display-based problem solving. Complex information processing: The impact of Herbert A. Simon, pages 319–341.
- Lund, H. H. and Miglino, O. (1998). Evolving and breeding robots. In *Evolutionary Robotics*, pages 192–210. Springer.
- Lund, H. H., Miglino, O., Pagliarini, L., Billard, A., and Ijspeert, A. (1998). Evolutionary robotics-a children's game. In Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, pages 154–158. IEEE.
- Nii, H. P. (1986). The blackboard model of problem solving and the evolution of blackboard architectures. *AI magazine*, 7(2):38.
- Peterson, J., Hager, G. D., and Hudak, P. (1999). A language for declarative robotic programming. In *Robotics and Automation*, 1999. Proceedings. 1999 IEEE International Conference on, volume 2, pages 1144–1151. IEEE.
- Pfeiffer Jr, J. J. (1998). Altaira: A rule-based visual language for small mobile robots. *Journal of Visual Languages and Computing*, 9(2):127–150.
- Secretan, J., Beato, N., D'Ambrosio, D. B., Rodriguez, A., Campbell, A., Folsom-Kovarik, J. T., and Stanley, K. O. (2011). Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary Computation*, 19(3):373–403.
- Stanley, K. O. and Miikkulainen, R. (2004). Competitive coevolution through evolutionary complexification. *J. Artif. Intell. Res.(JAIR)*, 21:63–100.
- Takagi, H. (1998). Interactive evolutionary computation: System optimization based on human subjective evaluation. In *IEEE Int. Conf. on Intelligent Engineering Systems (INES98)*, pages 17–19.
- Von Ahn, L. and Dabbish, L. (2004). Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326. ACM.
- Wilkinson, D. M. (2008). Strong regularities in online peer production. In *Proceedings of the 9th ACM conference on Electronic commerce*, pages 302–309. ACM.
- Woolley, A. W., Chabris, C. F., Pentland, A., Hashmi, N., and Malone, T. W. (2010). Evidence for a collective intelligence factor in the performance of human groups. *science*, 330(6004):686–688.
- Yu, L. and Nickerson, J. (2011). Generating creative ideas through crowds: An experimental study of combination. In *Thirty* Second International Conference on Information Systems.

Learning to Walk in Every Direction with the TBR-Learning algorithm

Antoine Cully^{1,2} and Jean-Baptiste Mouret^{1,2}

¹Sorbonne Universités, UPMC Univ Paris 06, UMR 722, ISIR, F-75005, Paris, France ²CNRS, UMR 7222, ISIR, F-75005, Paris, France mouret@isir.upmc.fr

Introduction

Legged robots are versatile machines that can outperform wheeled robots on rough terrain (Raibert, 1986), for instance in exploration or rescue missions. Their versatility is, however, tempered by their mechanical and control complexity, which makes them prone to mechanical damages and difficult to control robustly (Raibert, 1986; Bongard et al., 2006; Koos et al., 2013a). A promising way to compensate for these two weaknesses is to let robots discover *on their own* the best way to move in the current situation. A legged robot can thus cope with an unexpected terrain or with mechanical damages by learning a new walking gait (Bongard et al., 2006; Koos et al., 2013a), in the same way as animals can learn to limp with a sprained ankle.

Reinforcement learning (Kohl and Stone, 2004; Tedrake et al., 2005) and evolutionary algorithms (Zykov et al., 2004; Chernova and Veloso, 2004; Hornby et al., 2005) have been investigated to discover walking gaits for physical robots. Nevertheless, most of these investigations are limited to straight, forward walking, whereas a robot that only walks along a straight line is obviously unable to accomplish any mission. Only a handful of works deal with controllers able to turn or to change the walking speed. In these cases, controllers are successively evaluated on each possible direction (Mouret et al., 2006), or learned with an incremental process (Kodjabachian and Meyer, 1998). Compared to learning a simple controller, these two approaches significantly increase the learning time and the complexity of the search process.

In the present paper, we describe the Transferability-based Behavioral Repertoire Evolution (TBR-Evolution), a new learning algorithm that allows a robot to learn to walk in every direction in a single run of evolutionary algorithm. This algorithm combines the BR-Evolution algorithm (Cully and Mouret, 2013), which creates a behavioral repertoire in a single run, with the transferability approach (Koos et al., 2013b), which minimizes the number of evaluations on a physical robot when evolving controllers thanks to a simulator. A behavioral repertoire is a collection of simple controllers, where each of them reaches one position. An exter-

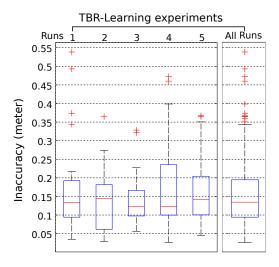


Figure 1: Inaccuracy of the controllers. The inaccuracy is measured as the distance between the endpoint reached by the *physical robot* and the one reached in simulation (30 repertoire's controllers are tested for each run). The results of the TBR-Evolution experiments are, for each run, separately pictured (Left) and also combined for an overall point of view (Right).

nal planing algorithm can thus successively pick up repertoire's controllers to generate a desired trajectory. Since TBR-Evolution does not depend on the controllers' architecture, it can be exploited in combination with most of the previous work about gait evolution.

Experiments

We evaluate the TBR-Evolution on a physical hexapod robot with 18 degrees of freedom (fig. 2B). The behaviors on the physical robot are assessed on-board thanks to a RGB-D sensor coupled with a state-of-the-art SLAM algorithm (Endres et al., 2012). For each TBR-Evolution experiment, a population of 100 solutions evolves during 3000 generations in simulation. After only 60 tests on the robot (one every 50 generations), the algorithm generates a collection with a

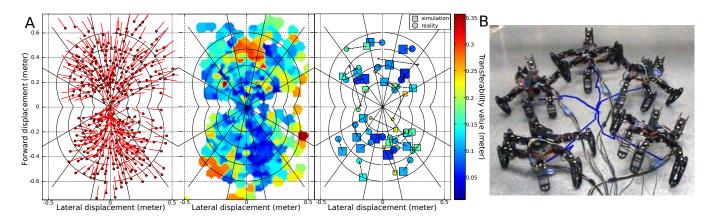


Figure 2: (A) Typical collection of controllers obtained with the TBR-Evolution algorithm. (Left) Endpoints in simulation of each controllers of the collection. The red lines are the final orientations of the robot. (Center) Estimated transferability map. The estimated transferability of each behavior of the collection is considered within a radius of 5 cm. (Right) Execution on the physical robot. we selected 30 controllers in the repertoire (square) and executed them on the physical robot (circles). The size and the color of the markers are proportional to their accuracy. (B) Illustration of 5 typical trajectories obtained with the TBR-Evolution.

median number of 375 controllers (min=353, max=394, five replications) allowing the robot to walk in every direction (fig. 2B). The periodical tests aim to drive the process to solutions that work similarly in simulation and in reality (i.e. crossing the reality gap, Koos et al. (2013b)). One of these collections is pictured in figure 2A.

The archive covers a large portion of the space around the robot (fig. 2A) and has an actual transferability value below 15 cm (difference between endpoints in the simulation and in reality. Fig. 1, left), which is consistent with the observations of the transferability map (Fig. 2, center) and not very large, taking into account the SLAM precision, the size of the robot and the looseness in the joints.

The source-code of our experiments and a supplementary video can be downloaded from: http://pages.isir.upmc.fr/evorob_db/

Acknowledgements

This work has been funded by the ANR Creadapt project (ANR-12-JS03-0009) and a DGA scholarship to A. Cully.

References

Bongard, J., Zykov, V., and Lipson, H. (2006). Resilient machines through continuous self-modeling. *Science*, 314(5802):1118– 1121

Chernova, S. and Veloso, M. (2004). An evolutionary approach to gait learning for four-legged robots. In *Proc. of IEEE/RSJ IROS*.

Cully, A. and Mouret, J.-B. (2013). Behavioral repertoire learning in robotics. In *Proc of GECCO*, pages 175–182. ACM.

Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., and Burgard, W. (2012). An evaluation of the RGB-D SLAM system. In *Proc. IEEE ICRA*.

Hornby, G., Takamura, S., Yamamoto, T., and Fujita, M. (2005). Autonomous evolution of dynamic gaits with two quadruped robots. *IEEE Trans. on Robotics*, 21(3):402–410.

Kodjabachian, J. and Meyer, J.-A. (1998). Evolution and development of neural controllers for locomotion, gradient-following, and obstacle-avoidance in artificial insects. *Neural Networks, IEEE Transactions on.*

Kohl, N. and Stone, P. (2004). Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proc. of IEEE ICRA*.

Koos, S., Cully, A., and Mouret, J.-B. (2013a). Fast damage recovery in robotics with the t-resilience algorithm. *The International Journal of Robotics Research*, 32(14):1700–1723.

Koos, S., Mouret, J.-B., and Doncieux, S. (2013b). The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Trans. on Evolutionary Computation*, pages 122–145.

Mouret, J.-B., Doncieux, S., and Meyer, J.-A. (2006). Incremental evolution of target-following neuro-controllers for flappingwing animats. *From Animals to Animats 9*.

Raibert, M. H. (1986). Legged robots. *Communications of the ACM*, 29(6):499–514.

Tedrake, R., Zhang, T., and Seung, H. (2005). Learning to walk in 20 minutes. In *Proc. of Yale workshop on Adaptive and Learning Systems*.

Zykov, V., Bongard, J., and Lipson, H. (2004). Evolving dynamic gaits on a physical robot. In *Proc. of GECCO, Late Breaking Paper*.

Investigating Modular Coupling of Morphology and Control with Digital Muscles

Jared M. Moore and Philip K. McKinley

Michigan State University, East Lansing, MI, USA 48824 moore112@msu.edu

Abstract

The musculoskeletal systems of animals are governed by a complex network of neurons that define both high- and lowlevel control. Individual joints are manipulated by multiple muscles acting as effectors for both movement and stabilization. We previously proposed a digital muscle model (DMM), where the morphological and control aspects of simulated joints evolve concurrently. The resulting solutions can provide insight into the evolution of natural organisms as well as possible designs for engineered systems. In this paper, we explore the integration of this model with an artificial neural network (ANN), focusing on the communication connections between the two. In the singly-connected strategy, a single ANN output is delivered to a joint; each constituent muscle responds to the signal according to an evolved function. In the individually-connected strategy, a unique ANN output is delivered to each simulated muscle. Results indicate that for low degree-of-freedom (DOF) robots, the individually-connected systems exhibit higher fitness than the singly-connnected systems. However, in larger DOF robots, the two strategies perform comparably, despite the fact that evolved ANNs for the singly-connected system are considerably simpler in terms of the number of connections in the network.

Introduction

Natural organisms demonstrate remarkable feats of locomotive agility, reaction, and maneuverability; the ability to run, jump, and accelerate has direct implications for survival. Complex neuromuscular and musculoskeletal systems underlie these abilities, which range from precise fine motor actions to purely strength based tasks (Gerritsen et al., 1998; Nishikawa et al., 2007). In nature, control and morphology evolve together. When applied to robots, this coevolutionary process has been effective in producing integrated controllers tuned to specific body configurations (Lipson and Pollack, 2000). Examples include bipedal locomotion (Paul and Bongard, 2001), gait development (Schramm et al., 2011), and wheeled robot navigation (Cliff et al., 1993). Typically, robotic joints comprise motor-driven actuators that receive commands directly from a high-level controller. In contrast, animal joints are composed of muscles, bones, and tendons that are tightly integrated with the somatic nervous system. Moreover, multiple muscles work together to control a single joint.

We previously developed a bioinspired abstract neuromuscular model called digital muscles (Moore and McKinley, 2014), which combines joint morphology and joint-level control. The orientation of muscle nodes around the joint, along with their (individual) responses to external signals, evolve concurrently. Evolving an abstract model has two major benefits. First, it is computationally efficient, enabling evolution of large populations over many generations. Second, the solutions discovered can be mapped into both detailed neuromuscular models of specific animals, providing insight into their evolution, as well as engineered joints based on motors. In addition, delegating aspects of control to the joint level potentially frees the high-level controller to focus on decision making. A set of evolutionary experiments with 3D animats demonstrated the evolution of effective gaits using only a simple sinusoid as the high-level control signal.

In this work we investigate combining this digital muscle model (DMM) with more complex high-level controllers, specifically, artificial neural networks (ANNs) evolved with the NEAT algorithm (Stanley and Miikkulainen, 2002). ANNs add a reactive control component that is not provided by the low-level DMM control framework. We explore two different approaches to interconnecting an ANN and a DMM-based joint: sending a common signal to all muscle nodes in the joint, or sending a separate signal to each muscle node in the joint. We refer to these configurations as *singly-connected* and *individually-connected*, respectively. Although the latter enables more fine-grained control of the joint, the number of ANN outputs increases rapidly with the complexity of the robot morphology.

Through a series of 10 treatments, we evaluate these two connection strategies in a ball balancing task and in the evolution of gaits for legged robots. We are particularly interested in how singly-connected systems, which enforce a modular coupling of ANN to DMM, thereby reducing the number of ANN outputs, perform against

individually-connected systems. Results show that in the ball balancing task, where the system has only a few joints, the individually-connected strategy outperforms the singly-connected strategy. However, in the quadruped and hexapod robot platforms, which have 8 and 12 joints respectively, the two control strategies perform comparably, despite the fact that the evolved ANNs in the singly-connected systems are considerably simpler, in terms of the number of connections in the networks. This result is consistent with theories of neural control in biological organisms where movement primitives in the spinal cord govern the coordination of multiple muscles (Giszter et al., 1993).

The contributions of this work are as follows. We first demonstrate a method to evolve aspects of control and morphology with an ANN and DMM. Second, evolved controllers exhibit effective behaviors with hybrid ANN/DMM controllers in two task environments. Third, this study provides insight into connection strategies between neural controllers and a digital muscle based joint-level control.

Background and Related Work

Evolutionary robotics (Floreano et al., 2008; Cliff et al., 1993; Nolfi and Floreano, 2000; Lipson, 2005) harnesses the process that has produced robust natural organisms and applies it to either physical or simulated robots. Evolved behaviors such as walking (Brooks, 1989), grasping (Bongard, 2010), and swimming (Sims, 1994) have been demonstrated. ANNs (Yao, 1999) are particularly amenable to evolutionary optimization, leading to effective locomotion in a robotic salamander (Ijspeert et al., 2005), bipeds (Reil and Husbands, 2002), and crawling robots (Inoue et al., 2007). Typically, these controllers produce high-level signals to govern the movement of each joint, such as specifying the desired angle of an actuator. In addition, a single controller often addresses multiple behaviors, from high-level decision making down to individual joint-level commands. In contrast, the movement of natural organisms is the result of complex interactions between an individual's neural and musculoskeletal systems (Wolpert and Ghahramani, 2000; Giszter et al., 1993).

This observation has led researchers to evolve morphology and control together. Bongard (2011) demonstrated the importance of morphology in the evolutionary process as a contributor to robustness in an individual. Embodiment (Pfeifer et al., 2007) of a controller within a morphology further emphasizes the coupled dynamics between the two. Paul and Bongard (2001) found that small changes to a robot's mass distribution have large effects on resultant gaits, leading to unique control/morphology pairs in evolved individuals. In this work, we investigate the joint-level coupling of morphology and control through the integration of ANN and DMM.

Simulated muscles have been investigated as effectors in a virtual robot platform by Lessin et al. (2013). Actuation occurs when a spring constant is changed, contracting the simulated muscle, subsequently moving the attached limbs. Geijtenbeek et al. (2013) demonstrated virtual muscle driven bipeds, capable of walking on both flat and varying surfaces. Muscles emulate physical properties defining attachment points, contraction paths, and actuation parameters determined through the optimization process. The gaits derived from this model feature realistic movements for the virtual creatures. These models emulate physical muscles directly in the simulated platform, whereas the proposed DMM does not consider muscles to be effectors modeled directly in a robot. Instead, the model focuses on the position and activation of muscles. This enables application of the DMM to motor based joints in traditional robots while contributing to the study of biological organisms.

Focusing on modularity in controller design may lead to more robust behaviors by introducing smaller, functional units for specific tasks (Pasemann et al., 2001; Brooks, 1986). Indeed, Doncieux and Meyer (2004) have shown that it may be difficult to develop complex control strategies without structural modularity in ANNs. Currently, modular approaches have focused primarily on the ANNs themselves. The DMM defines aspects of both morphology and control. In this paper, we investigate coupling strategies between ANN and DMM.

Digital Muscle Model

In (Moore and McKinley, 2014), we proposed the digital muscle model, a framework to model properties of physical muscles with the ability to map commands to conventional servo-driven robots. The DMM draws inspiration from natural organisms whose muscles provide the power necessary for movement. Working in antagonistic pairs, muscles facilitate flexion and extension of joints. Figure 1 shows a hinge joint controlled by a DMM muscle group.

Conceptually, the DMM organizes muscle *nodes*, analogous to biological muscles, into muscle groups at the joint level. The number of muscle nodes in a group is variable; in this study each muscle group consists of four muscle nodes. Each muscle node "contracts" in response to an input signal with the specific response determined by an internal activation function.

Muscle Node Structure. The position and activation of a digital muscle node govern its behavior. In this study, the activation function is a Gaussian with evolvable parameters: μ (center), σ^2 (spread) and α (magnitude). Similar to biological muscles, the nodes exert only contracting forces on the limbs. Direction of pull for a contracting node is determined by a node's position. The positions of nodes are evolvable, allowing for refinement of joint range of motion.

Calculation of Movement Commands. Figure 2 shows the activation functions of four nodes in a muscle group over

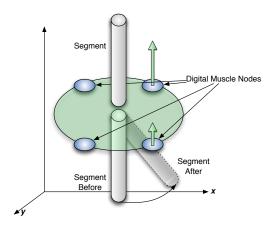


Figure 1: A joint in an animat is controlled by a muscle group comprising a set of muscle nodes. Each node is located on a 2D plane around the joint. When activated, nodes exert a pulling force on the limb segment, drawing it toward the node's position. Joint movement is calculated by aggregating the pulling forces of the muscle nodes.

the input range of [-1.0, 1.0]. Together with the positions of the four nodes, seen in Figure 1, they produce the commands seen in Figure 3. These two commands specify the movement of a single 2 DOF joint in the robot.

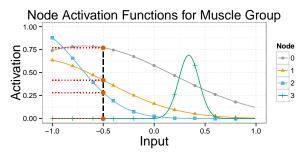


Figure 2: Activation functions of four nodes in a muscle group. At input value -0.5, the outputs correspond with the red circles intersecting the vertical line.

ANN/Muscle Model Integration. In our original study (Moore and McKinley, 2014), evolved quadrupeds exhibited successful gaits with DMM controllers, despite being driven by a simple sinusoidal signal. In this study, we connect the DMM to an ANN, creating a two-tiered control method. Figure 4 illustrates the two different strategies for connecting ANN to the DMM. The top figure shows the singly-connected strategy, where each muscle group is connected to one ANN output. The same ANN signal is passed to each node in a muscle group. Movement of the joint is then determined by aggregating the individual responses of each node to the signal. The bottom figure shows the individually-connected strategy, where each

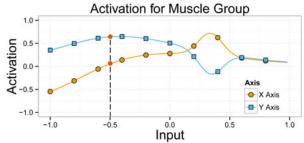


Figure 3: Joint movements are calculated by aggregating the activations and spatial position of each node in a muscle group. The activation functions from Figure 2 and spatial position of nodes seen in Figure 1 create the x and y axis outputs for a 2 DOF joint seen here. Nodes are positioned radially around a joint at 45°, 135°, 225°, and 315°.

node of a muscle group is connected to a unique ANN output. Here, the position and activation of a muscle node still determine its movement, but individual muscle nodes in a muscle group receive different commands from the ANN. Hence in this example, individually-connected ANNs require 4 times as many outputs as the singly-connected ANNs.

Methods

In this section we introduce specifics of ball balancing and gait evolution and discuss the evolutionary setup used in both tasks.

Ball Balancing Task. In the first task, an articulated platform is required to balance a ball dropped onto the center of the platform. Figure 5 shows the one-, two-, or three-segment arms with a platform at the top. The arm is affixed to the ground. Each joint has 2 DOF with $\pm 90^{\circ}$ of rotation on each axis. A ball is dropped from slightly above the platform to start a simulation. Upon contact with the platform, the ball is given a strong push in one of eight possible directions (N,NE,E,SE,S,SW,W,NW), attempting to knock it to the ground. Each simulation lasts for 20 seconds or until the ball contacts the ground. Fitness is assessed as the time interval between the ball's first contact with the platform and the last time of contact.

Six treatments are conducted in the ball balancing task. The first three treatments (A1,A2,A3) have singly-connected ANN/DMM controllers. Treatment A1 has a single joint connecting an arm segment to the platform. Treatment A2 adds an additional joint and arm segment, while the arm in Treatment A3 contains three joints. The other three treatments (B1,B2,B3) use individually-connected ANN/DMM configurations, with 1-, 2-, and 3-joint arms respectively. Each individual is evaluated in 5 environments randomly drawn from the 8 push directions mentioned previously. An

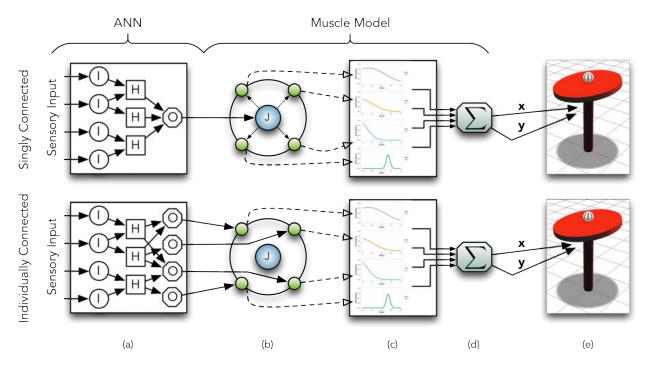


Figure 4: Examples illustrating the two connection strategies tested in this study: (top) singly-connected and (bottom) individually-connected. Interaction between the ANN and each DMM-based joint proceeds as follows: (a) The ANN receives input from sensors and produces output(s), 1 for a singly-connected joint and 4 for an individually-connected joint. (b) For a singly-connected joint, the same ANN output signal is distributed to each of 4 muscle nodes. For an individually connected joint, each muscle node receives its own signal directly from the ANN. (c) The position and activation function of each muscle node determines its response to the incoming signal. (d) The responses of the muscle nodes are aggregated and (e) passed to the platform.

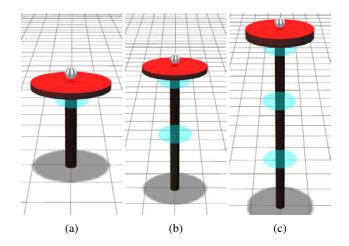


Figure 5: The platforms have either one, two or three actuated joints. Each joint is a 2 DOF hinge, allowing for pivoting on the x and z axes, with y being the radial axis of the arm. One joint connects the arm to the platform in each configuration with the others indicated by the transparent ellipsoids.

individual must react accordingly to the push, keeping the ball on the platform for as long as possible. Fitness is the average of the 5 simulations.

ANNs for the ball balancing task have nine to thirteen inputs. Three inputs represent the xyz distance between the platform and ball, including a sign to indicate direction, three represent the xyz linear velocity of the ball; a bias is also included. The final inputs comprise the relative angles [-1.0, 1.0] for the two axes in each 2 DOF joint. Inputs thus range from 9 to 13, depending on the number of arm segments in the platform.

Legged Gait Evolution. The second task applies the ANN/DMM to locomotion in legged robots. We conducted an additional four treatments (A8,B8,A12,B12) in an 8-joint quadruped and 12-joint hexapod to assess how modular connection strategies perform in higher DOF robots. Figure 6 shows the 8-joint quadruped robot and 12-joint hexapod robot used in this task. Two treatments (A8,A12) are conducted with singly-connected ANN/DMM controllers, and two treatments (B8,B12) with individually-connected ANN/DMM controllers. In Treatment A8, ANNs have 8 outputs, while in Treatment B8, the ANN has 32 outputs.

Accordingly, Treatment A12 has 12 outputs, and Treatment B12 has 48 outputs. Individuals are evaluated based on the total distance traveled in 10 seconds of simulation time on a high friction surface. Inputs to the ANN for both configurations include a sinusoidal signal, a bias, touch sensors for the feet, and 2 angle sensors per joint.

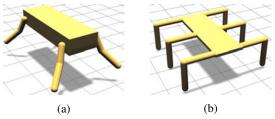


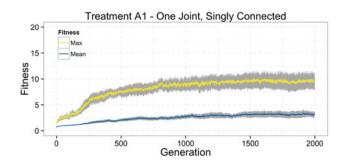
Figure 6: The digital quadruped (8 joints) and hexapod (12 joints) used in legged gait evolution. Each leg has two 2 DOF joints (hip and knee).

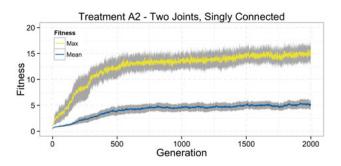
Evolutionary Setup. Populations comprise 100 individuals and are evolved for 2000 generations. We conduct 20 replicate runs per treatment, each with a unique starting seed. ANNs are evolved using the NEAT algorithm (Stanley and Miikkulainen, 2002). DMMs are paired with an ANN through a genome identifier. Thus, crossover between two DMMs is applied only when the two associated ANNs are recombined. Mutation is applied per generation to individual DMMs with a 5% chance per parameter; each joint has four nodes with four parameters each.

Results

Ball Balancing Performance. Figures 7 and 8 show the fitnesses of the 6 treatments from the ball balancing task. Here, fitnesses correspond to how long (seconds), on average, an individual balances the ball. Simulations last 20s. Treatments A1, A2, and A3 have varying success balancing the ball across replicates, however, no replicate run evolves an individual capable of balancing the ball for the full 20s in all 8 situations. Surprisingly, Treatment A3 fails to evolve any meaningful behavior. Apparently, the evolutionary process is unable to develop control strategies to coordinate the three joints.

In contrast, Treatments B1, B2, and B3 all produce effective behaviors, as shown in Figure 8. Furthermore, the population average fitnesses are also significantly higher than the singly-connected treatments. The evolutionary trajectories of the three treatments are similar, although it appears to take longer to evolve solutions for more joints. During validation against the full 8 push directions, all but 1 of the replicates from Treatment B1, every replicate in Treatment B2, and all but 2 replicates in Treatment B3 evolved individuals capable of handling all conditions.





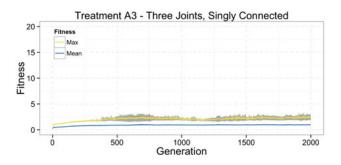
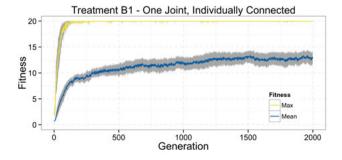
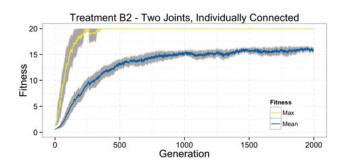


Figure 7: Mean fitnesses from 20 replicate runs for singly-connected individuals. Shaded regions represent 95% confidence intervals.

Evolved ANN Complexity. Figure 9 shows the relative complexity of the evolved ANNs for each of the treatments, defined as the number of connections between nodes in the ANNs. For Treatment A, the most effective networks arise in Treatment A2, which coincidentally have the highest ratio of performance to network size. Larger networks result in higher fitness values for Treatment A2. Conversely, Treatments B1, B2, and B3 all exhibit less complex networks. The connection strategy between ANN and DMM appears to be a key indicator of performance in this task. The singly-connected strategy is unable to evolve a solution capable of addressing all 8 push directions. Treatment B is able to address the task, with ANNs having individual connections to the muscle nodes. Presumably, this allows for finer control of the joints in these low DOF morphologies. Even with the





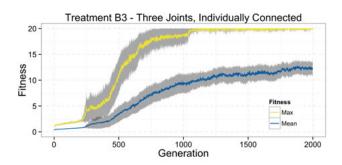
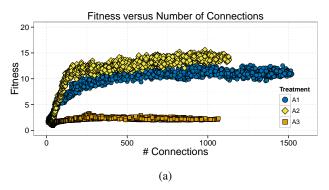


Figure 8: Mean fitnesses from 20 replicate runs for individually-connected individuals. Shaded regions represent 95% confidence intervals.

increased output space required in the ANNs of Treatment B, the number of connections in the best evolved networks is half that of Treatment A.

Apparently, this modular connection strategy in Treatment A necessitates the evolution of more complex networks in all three platform configurations. The reduced connectivity in Treatment A requires the evolution of coordination between muscle node activation functions for a single input signal from the ANN at the joint-level. Next, we investigate whether these results scale to legged robots with 8- and 12-joints.

Legged Gait Evolution. Figures 10a and 10b, respectively, plot the fitnesses for the quadruped and hexapod runs. Unlike the ball balancing task, individually-connected con-



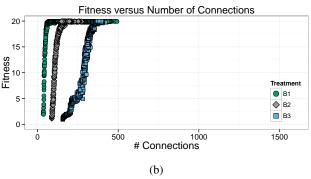
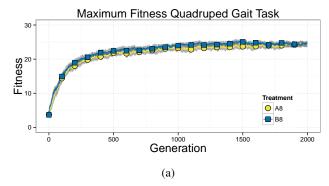


Figure 9: Number of connections between nodes in the ANN versus fitness across the treatments. (a) Evolved networks in Treatment A exhibit between 500 and 1500 connections in the top 10% of performers. (b) Evolved networks from Treatment B exhibit 100 and 500 connections in the top 10% of performers. In addition, the B treatments are all able to address the ball balancing task.

trollers in Treatment B8 only slightly outperform singlyconnected individuals in Treatment A8, with the 95% confidence intervals overlapping. Performance is nearly identical between Treatments A12 and B12. Figure 11 shows the difference in ANN complexity, with the singly-connected strategy producing less complex networks in contrast to the results seen in the ball balancing task. Figure 12 shows the resulting number of connections in the evolved ANNs for Treatments A12 and B12. In this case the network complexity further diverges, although both strategies exhibit effective walking gaits. That A treatments achieve performance similar to the B treatments, using less complex ANNs, suggests that the singly-connected strategy is an effective controller in these higher DOF robots. Reductions in network complexity become more important as the DOF in a robot increases. In robots with greater DOF, ANN complexity can impact controller performance, making compact networks more desirable.



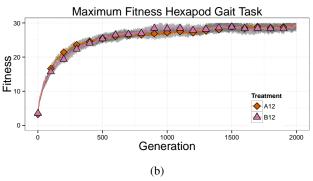


Figure 10: (a) The best performing individuals from Treatments A8 and B8 in a quadruped gait evolution task. (b) The best performing individuals from Treatments A12 and B12 in a hexapod gait evolution task. Each line represents the average of 20 replicate runs with shaded areas indicating 95% confidence intervals.

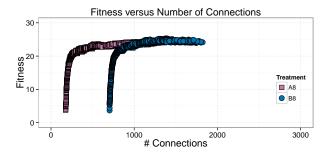


Figure 11: Number of connections between nodes in the ANN versus fitness for the quadruped platform. Here, the evolved ANNs for Treatment A8 are less complex than those of Treatment B8, even though performance is similar, suggesting that as the number of joints increases, a singly-connected control strategy performs comparably with a less complex ANN.

Conclusions

In biological organisms, multiple muscles cooperate to realize movement of joints. The digital muscle model provides a computationally efficient means to evolve joint-level con-

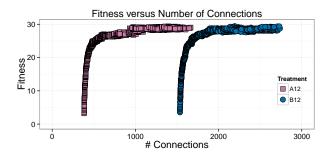


Figure 12: Number of connections between nodes in the ANN versus fitness for the hexapod platform. Evolved ANNs for Treatment A12 contain fewer connections than those of Treatment B12.

trol in 3D animats. The model captures fundamental properties of natural muscles while remaining abstract enough to apply to robotic joints. In this paper, we have examined the integration of a high-level ANN controller with low-level DMM-based joints to realize effective behavior in robots. In the ball balancing task, where the system has up to three joints, individually-connecting ANN outputs to muscle nodes produces higher fitness than singly-connected systems. When evolving gaits in the 8-joint quadruped, however, individually- and singly-connected ANN/DMM controllers perform comparably, with evolution producing considerably simpler ANNs in the latter case. Results for the 12-joint hexapod robot are similar, with even greater disparity in network complexity. In the future, we plan to investigate the coupling of these two strategies in higher DOF robots and for more complex tasks. In addition, we plan to investigate the effect of enforced symmetries and indirect mappings, which were intentionally left out in this work in order to focus on interaction between the ANN and DMM.

Acknowledgements

The authors gratefully acknowledge the contributions and feedback provided by Anthony Clark, Xiaobo Tan, Anne Gutmann, Craig McGowan, and members of the BEACON Center at Michigan State University. This work was supported in part by National Science Foundation grants CNS-1059373, CNS-0915855, and DBI-0939454, and by a grant from Michigan State University.

References

Bongard, J. (2011). Morphological change in machines accelerates the evolution of robust behavior. *Proceedings of the National Academy of Sciences*, 108(4):1234–1239.

Bongard, J. (2010). The utility of evolving simulated robot morphology increases with task complexity for object manipulation. *Artificial Life*, 16(3):201–223.

- Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23.
- Brooks, R. A. (1989). A robot that walks; emergent behaviors from a carefully evolved network. *Neural Computation*, 1(2):253–262.
- Cliff, D., Husbands, P., and Harvey, I. (1993). Explorations in Evolutionary Robotics. *Adaptive Behavior*, 2(1):73–110.
- Doncieux, S. and Meyer, J.-A. (2004). Evolving modular neural networks to solve challenging control problems. In *Proceedings of the Fourth International ICSC Symposium on Engineering of Intelligent Systems (EIS 2004)*, pages 1–7, Madeira, Portugal.
- Floreano, D., Husbands, P., and Nolfi, S. (2008). Evolutionary Robotics. In *Handbook of Robotics*. Springer Verlag, Berlin.
- Geijtenbeek, T., van de Panne, M., and van der Stappen, A. F. (2013). Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics*, 32(6):206:1–206:11.
- Gerritsen, K. G., van den Bogert, A. J., Hulliger, M., and Zernicke, R. F. (1998). Intrinsic muscle properties facilitate locomotor control a computer simulation study. *Motor Control*, 2(3):206–220.
- Giszter, S. F., Mussa-Ivaldi, A., and Bizzi, E. (1993). Convergent force fields organized in the frog's spinal cord. *Journal of Neuroscience*, 13(2):467–491.
- Ijspeert, A., Crespi, A., and Cabelguen, J. (2005). Simulation and robotics studies of salamander locomotion. *Neuroinformatics*, 3(3):171–195.
- Inoue, K., Sumi, T., and Ma, S. (2007). CPG-based control of a simulated snake-like robot adaptable to changing ground friction. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1957 –1962, San Diego, California, USA.
- Lessin, D., Fussell, D., and Miikkulainen, R. (2013). Openended behavioral complexity for evolved virtual creatures. In *Proceedings of the 2013 ACM Genetic and Evolutionary Computing Conference*, pages 335–342, Amsterdam, Netherlands. ACM.
- Lipson, H. (2005). Evolutionary robotics and open-ended design automation. In *Biomimetics*, pages 129–155. CRC Press.
- Lipson, H. and Pollack, J. B. (2000). Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974–978.
- Moore, J. M. and McKinley, P. K. (2014). (Accepted to Appear) Evolving joint-level control with digital muscles. In *Proceedings of the 2014 ACM Genetic and Evolutionary Computing Conference*, Vancouver, BC, Canada. ACM.
- Nishikawa, K., Biewener, A. A., Aerts, P., Ahn, A. N., Chiel, H. J., Daley, M. A., Daniel, T. L., Full, R. J., Hale,

- M. E., Hedrick, T. L., Lappin, A. K., Nichols, T. R., Quinn, R. D., Satterlie, R. A., and Szymik, B. (2007). Neuromechanics: an integrative approach for understanding motor control. *Integrative and Comparative Biology*, 47(1):16–54.
- Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics:* The Biology, Intelligence and Technology of Self-Organizing Machines. The MIT Press.
- Pasemann, F., Steinmetz, U., Hulse, M., and Lara, B. (2001). Robot control and the evolution of modular neurodynamics. *Theory in Biosciences*, 120(3-4):311–326.
- Paul, C. and Bongard, J. C. (2001). The road less travelled: Morphology in the optimization of biped robot locomotion. In *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 226 – 232, Maui, Hawaii, USA.
- Pfeifer, R., Lungarella, M., and Iida, F. (2007). Self-Organization, Embodiment, and Biologically Inspired Robotics. *Science*, 318(5853):1088–1093.
- Reil, T. and Husbands, P. (2002). Evolution of central pattern generators for bipedal walking in a real-time physics environment. *IEEE Transactions on Evolutionary Computation*, 6(2):159–168.
- Schramm, L., Jin, Y., and Sendhoff, B. (2011). Emerged coupling of motor control and morphological development in evolution of multi-cellular animats. In *Proceedings of the 10th European Conference on Advances in Artificial Life: Darwin Meets von Neumann*, pages 27–34, Budapest, Hungary. Springer-Verlag.
- Sims, K. (1994). Evolving virtual creatures. In *Proceedings* of the 21st Annual Conference on Computer Graphics and Interactive Techniques, pages 15–22.
- Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.
- Wolpert, D. M. and Ghahramani, Z. (2000). Computational principles of movement neuroscience. *Nature Neuroscience*, 3 Suppl:1212–1217.
- Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447.

Abstract of: Fast Damage Recovery in Robotics with the T-Resilience Algorithm

Sylvain Koos^{1,2}, Antoine Cully^{1,2} and Jean-Baptiste Mouret^{1,2}

Sorbonne Universités, UPMC Univ Paris 06, UMR 722, ISIR, F-75005, Paris, France CNRS, UMR 7222, ISIR, F-75005, Paris, France mouret@isir.upmc.fr

Damage recovery is critical for autonomous robots that need to operate for a long time without assistance. Most current methods are complex and costly because they require anticipating each potential damage in order to have a contingency plan ready and diagnosis procedures.

An alternative line of thought is to *let the robot learn on its own* the best behavior for the current situation. If the learning process is open enough, then the robot should be able to discover new compensatory behaviors in situations that have not been foreseen by its designers. Classic reinforcement learning algorithms are hard to apply to low-level robotic problems (Togelius et al., 2009), but evolutionary algorithms (EAs) are good candidates to find original solutions because they can optimize in the continuous domain and work on the structure of controllers (e.g. neural networks).

When evolving controllers for robots, EAs are reported to require many hundreds of trials on the robot and to last from two to tens of hours (e.g. (Hornby et al., 2005; Yosinski et al., 2011)). These EAs spend most of their running time in evaluating the quality of controllers by testing them on the target robot. Since, contrary to simulation, reality cannot be sped up, their running time can only be improved by finding strategies to evaluate fewer candidate solutions on the robot.

By first learning a self-model for the robot, then evolving a controller with this simulation, Bongard et al. (Bongard et al., 2006) designed an algorithm for resilience that makes an important step in this direction. Nevertheless, this algorithm has a few important shortcomings. First, actions and models are undirected: the algorithm can "waste" a lot of time to improve parts of the self-model that are irrelevant for the task. Second, the diagnosis may be wrong, which leads to a useless contingency plan. Third, there is often a "reality gap" between a behavior learned in simulation and the same behavior on the target robot (Jakobi et al., 1995), but nothing is included in Bongard's algorithm to prevent such gap to happen: the controller learned in the simulation may not work well on the real robot, even if the self-model is accurate.

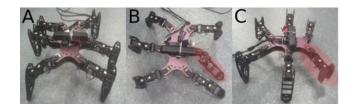


Figure 1: (A) The hexapod robot is not damaged. (B) The left middle leg is no longer powered. (C) The terminal part of the front right leg is shortened by half.

Our algorithm is inspired by the "transferability approach" (Koos et al., 2013b), whose original purpose is to cross the "reality gap" that separates behaviors optimized in simulation to those observed on the target robot. The main proposition of this approach is to make the optimization algorithm aware of the limits of the simulation. To this end, a few controllers are transferred during the optimization and a regression algorithm (here a SVM) is used to approximate the function that maps behaviors in simulation to the difference of performance between simulation and reality. To use this approximated transferability function, the singleobjective optimization problem is transformed into a multiobjective optimization in which both performance in simulation and transferability are maximized. This optimization is performed with a a multi-objective evolutionary algorithm (NSGA-II, Deb et al. (2002)).

The same concepts can be applied to design a fast adaptation algorithm for resilient robotics, leading to a new algorithm that we called "T-Resilience" (for Transferability-based resilience). If a damaged robot embeds a simulation of itself, then behaviors that rely on damaged parts will not be transferable: they will perform very differently in the self-model and in reality. During the adaptation process, the robot will thus create an approximated transferability function that classifies behaviors as "working as expected" and "not working as expected". Hence the robot will possess an "intuition" of the damages but it will not explicitly represent or identify them. By optimizing both the transferability and

This paper is an extended abstract of Koos et al. (2013a).

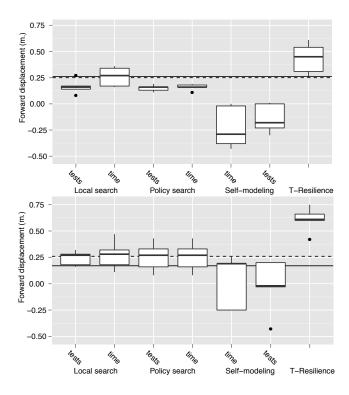


Figure 2: Performances (distance covered in 3 seconds) obtained in case B (top) and C (bottom). On each box, Each algorithm has been run 5 times and distances are measured using the external motion capture system. Except for the T-Resilience, the performance of the controllers found after about 25 transfers (tests) and after about 20 minutes (time) are depicted (all T-Resilience experiments last about 20 minutes and use 25 transfers). The horizontal lines denote the performances of the reference gait, according to the external (dashed line) and to the internal (solid line) measurement.

the performance, the algorithm will look for the most efficient behaviors among those that only use the reliable parts of the robots. The robot will thus be able to sustain a functioning behavior when damage occurs by learning to avoid behaviors that it is unable to achieve in the real world. Besides this damage recovery scenario, the T-Resilience algorithm opens a new class of adaptation algorithms that transfers most of the adaptation time from real experiments to simulations of a self-model.

Experiments

We evaluate the T-Resilience algorithm on an 18-DOFs hexapod robot that needs to adapt to motor failures and broken legs (figure 1); we compare it to stochastic local search (Hoos and Stützle, 2005), policy gradient (Kohl and Stone, 2004) and Bongard's algorithm (Bongard et al., 2006). The behavior on the real robot is assessed on-board thanks to a RGB-D sensor coupled with a state-of-the-art SLAM algorithm (Endres et al., 2012). For each experi-

ment, a population of 100 controllers is optimized for 1000 generations. Every 40 generations, a controller is randomly selected in the population and transferred on the robot.

Using only 25 tests on the robot and an overall running time of less than one hour on a recent laptop, T-Resilience consistently leads to substantially better results than the other approaches (figure 2).

Acknowledgements

This work has been funded by the ANR Creadapt project, (ANR-12-JS03-0009) and a DGA scholarship to A. Cully.

References

- Bongard, J., Zykov, V., and Lipson, H. (2006). Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121.
- Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. on Evol. Comp.*, 6(2):182–197.
- Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., and Burgard, W. (2012). An evaluation of the RGB-D SLAM system. In *Proc. of ICRA*.
- Hoos, H. H. and Stützle, T. (2005). *Stochastic local search: Foundations and applications*. Morgan Kaufmann.
- Hornby, G., Takamura, S., Yamamoto, T., and Fujita, M. (2005). Autonomous evolution of dynamic gaits with two quadruped robots. *IEEE Trans. on Robotics*, 21(3):402–410.
- Jakobi, N., Husbands, P., and Harvey, I. (1995). Noise and the reality gap: The use of simulation in evolutionary robotics. *Proc. of ECAL*, pages 704–720.
- Kohl, N. and Stone, P. (2004). Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proc. of ICRA*, volume 3, pages 2619–2624. IEEE.
- Koos, S., Cully, A., and Mouret, J.-B. (2013a). Fast damage recovery in robotics with the t-resilience algorithm. *The International Journal of Robotics Research*, 32(14):1700–1723.
- Koos, S., Mouret, J.-B., and Doncieux, S. (2013b). The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Trans. on Evol. Comp.*, 17:122–145.
- Togelius, J., Schaul, T., Wierstra, D., Igel, C., and Schmidhuber, J. (2009). Ontogenetic and phylogenetic reinforcement learning. *Kuenstliche Intelligenz*, 23:30–33.
- Yosinski, J., Clune, J., Hidalgo, D., Nguyen, S., Zagal, J., and Lipson, H. (2011). Evolving Robot Gaits in Hardware: the HyperNEAT Generative Encoding Vs. Parameter Optimization. *Proc. of ECAL*, pages 1–8.

Making MONEE

Evert Haasdijk¹, Nicolas Bredeche^{2,3} and A.E. Eiben¹

¹Computer Science Department, VU University Amsterdam, Amsterdam, Netherlands
 ²Sorbonne Universités, UPMC Univ Paris 06, UMR 7222, ISIR, F-75005, Paris, France
 ³CNRS, UMR 7222, ISIR, F-75005, Paris, France
 e.haasdijk@vu.nl

Abstract

This is an extended abstract of a recent PLoS ONE paper (Haasdijk et al., 2014) in which we introduce the MONEE paradigm as a method of combining open-ended (to deal with the environment) and task-driven (to satisfy user demands) adaptation of robot controllers through evolution.

Introduction

Evolutionary Robotics is concerned with the use of Evolutionary Algorithms to develop robot controllers and sometimes robotic hardware. The majority of related work is inherently goal-driven and uses evolution to optimise task performance of the robots for some clearly defined set of tasks. A few papers have investigated a more ALife flavoured approach where evolution is environment-driven, i.e., the chances for reproduction and survival are not based on user defined task performance.

In recent works, we argued that balancing evolution between environment-driven adaptation and task-driven optimisation is essential for self-sufficient robot collectives in environments where human supervision is impossible, too expensive, or limited. To implement such a combination, we recently introduced an algorithmic framework dubbed Monee (Multi-Objective aNd open-Ended Evolution) (Haasdijk et al., 2014). The main idea is to exploit the fact that evolutionary methods have two basic selection mechanisms and to use these in different roles: survivor selection driven by the environment and parent selection based on task-performance.

In the following, we review briefly the mechanisms at work in MONEE, and provide evidence that it can actually perform adaptation to the environment *and* objective-driven evolutionary optimisation.

MONEE

The proof of concept for MONEE is based on mEDEA—an ALife flavoured system introduced by Bredeche et al. (2012). In this system, robot controllers evolve without an explicit fitness measure in a distributed fashion, i.e., without a central 'oracle' orchestrating selection and reproduction. Robot controllers can procreate by transmitting their

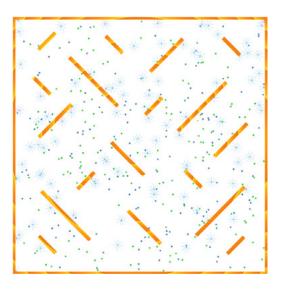


Figure 1: **Experiment screenshot.** Robots are shown as small circles with sensor beams indicated. Pucks are shown as small blue and green squares. The shaded orange rectangles indicate arena walls and obstacles.

genome to other robots directly. However, sending and receiving genomes are regulated by distinguishing two phases in each robots lifecycle.

- **Life phase**: robots controllers have a fixed lifetime during which they perform their actions; moving about, foraging, and sending around their genomes.
- Rebirth phase: when its lifetime ends, a robot enters the rebirth phase and becomes an 'egg': a stationary receptacle for genomes that are transmitted by passing live robots. This rebirth phase also lasts a fixed amount of time, and once this has passed, the egg selects parents from the received genomes to create a new controller. The robot then reverts to the 'life' role with this new controller

Note that the more eggs a robot inseminates, the more chances it has for procreation. Experiments with mEDEA

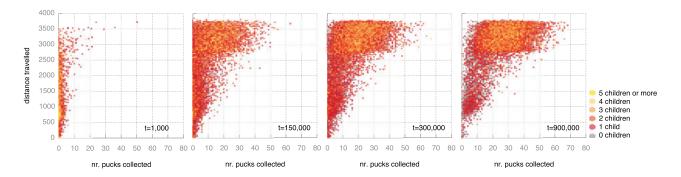


Figure 2: **Offspring count vs distance travelled and number of pucks collected.** From left to right, plots shown for time intervals at 1,000, 150,000, 300,000 and 900,000 ticks. Initially, distance travelled and number of pucks collected have little influence on an individual's fecundity. As time progresses, the influence of travelling distances and collecting pucks becomes progressively pronounced.

demonstrated a trend towards increased mobility as an evolutionary response to these environmental conditions.

MONEE adds task-driven parent selection by considering credits amassed by robots when they perform tasks (e.g. a gathering task in our experiments). When a robot inseminates an egg, it passes the current credit counts along with the genome and the egg uses that information to select parents when it revives, reminiscent of parental investment schemes (although a parent does not actually invest when impregnating an egg because the credits are not *transferred* but *copied*, at no cost to the parent). Thus, the credits relate task performance to reproductive success.

One particular advantage of MONEE is that when two concurrent tasks must be addressed, it is possible to tune selection pressure on-the-fly to effectively balance between the two tasks, favouring the least sought task. This market mechanism is reminiscent of fitness sharing in the sense that it also reappraises fitness, favouring tasks that are less commonly tackled by robots in the population, even though robots' behaviours are under scrutiny, rather than their genetic make-up. Figure 1 illustrates such a setup, where robots have to collect two types of pucks.

Combining environment-driven adaptation and objective-driven optimisation

In this puck-gathering scenario, there are two obvious determinants of selection pressure, i.e., two factors that determine the likelihood of a robot producing offspring. One is distance travelled. This is not an explicit objective, but it is implied by the fact that robots must come within communication range of eggs: robots that move about a lot have higher chances of meeting eggs and therefore procreate at a higher rate than robots that move little. The second factor is the task performance that we explicitly introduced: when reviving, an egg selects a parent (using binary tournament) for the new controller based on the number of pucks collected. For a qualitative view of the importance of these two

selective forces, consider Fig. 2. It plots the combined individuals of 64 runs in four 5,000 clock-tick intervals. Each individual is indicated by a small circle, of which the colour indicates the number of offspring for that individual. The position of the circle shows the number of pucks that individual collected (horizontal axis) and the total distance covered (vertical axis, in pixels in the simulated environment) during its lifetime. For full details of the experimental setup, see Haasdijk et al. (2014).

Initially (left-most panel), there is little variation in terms of pucks collected (the dots are concentrated between 0 and 5 pucks collected). Individuals with high offspring counts are found across the range of distance travelled with a slight concentration between 500-1,000. This indicates that there is little pressure towards movement or collecting pucks at this point. As evolution progresses, at t = 150,000, having offspring becomes contingent on travelling greater distances and collecting pucks; most robots travel substantial distances between 3,000 and 4,000 pixels. At t = 900,000, almost no individuals with more than one child that have travelled less than 2500 pixels. The two right-hand panels show that the number of collected pucks becomes increasingly important. These results indicate that robot behaviour initially adapts to the environment, evidenced by the initial differentiation in distance travelled. As evolution progresses and almost all robots travel substantial distances, the number of pucks collected becomes the dominant factor in determining the chances of producing offspring.

References

Bredeche, N., Montanier, J.-M., Liu, W., and Winfield, A. F. (2012). Environment-driven distributed evolutionary adaptation in a population of autonomous robotic agents. *Mathematical and Computer Modelling of Dynamical Systems*, 18(1):101–129.

Haasdijk, E., Bredeche, N., and Eiben, A. E. (2014). Combining environment-driven adaptation and task-driven optimisation in evolutionary robotics. *PLoS ONE*. In press.

On bootstrapping sensori-motor patterns for a constructivist learning system in continuous environments

Sébastien Mazac^{1,2}, Frédéric Armetta² and Salima Hassas²

¹ubiant, Lyon, France
 ²Universitée Lyon 1, LIRIS, UMR5205, F-69622, France sebastien.mazac@liris.cnrs.fr

Abstract

The theory of cognitive development from Jean Piaget (1923) is a constructivist perspective of learning that has substantially influenced cognitive science domain. Indeed it seems that constructivism is a possible trail in order to overcome the limitations of classical techniques stemming from cognitivism or connectionism and create autonomous agents, fitted with strong adaptation ability within their environment, modelled on biological organisms. Potential applications concern intelligent agents in interaction with a complex environment, with objectives that cannot be predefined. There are numerous interesting works in developmental robotics going in this direction. In this work we investigate the application of these principles to a close domain: Ambient intelligence, which is extremely challenging but which also presents interesting aspects to exploit, like the participation of human users. From the perspective of a constructivist theory, the learning agent has to build a representation of the world that relies on the learning of sensori-motor patterns starting from its own experience only. This step is difficult to set up for systems evolving in continuous environments, using raw data from sensors without a priori modelling, primarily because they face a bootstrap problem. In this paper we address this particular issue and propose a decentralized approach based on a multi-agent framework, where the system's representations are constructed through a self-organization process that handles the dynamics between experience discretization and learning.

Introduction

Embodiment is a paradigm in cognitive science and AI considering that *cognition* arises from the *interaction* between an *agent* and its *environment*. Contrary to computational theory of mind, an agent has to be embodied and situated, and does not have a predefined *representation*. On the other hand an agent (for example a robot) must be able to construct its own representation of the regularities through its interaction with the environment, whatever its body's capacities and the environment where it is situated. More generally, embodiment is embraced by constructivist theory of cognition, as explained by Ziemke (2001). Constructivism in cognitive science is bound to constructivist epistemology. Particularly Riegler (2001) explains why the *radical con*-

structivism proposed by Von Glasersfeld (1984) is necessary to go further in the understanding of cognition mechanisms. There is a growing interest in AI for constructivist theories, for example such as *Enaction* (Varela et al. (1991)) stemming from the biology field, or constructivist theory of learning (Piaget (1954)) coming from the psychology field. While it is true that constructivist approaches of learning in AI are usually still theoretical, there are also more practical domains that take inspiration from these concepts such as developmental robotics. This article addresses some fundamental issues related to the application of constructivist learning methods to continuous environments. In the following, we will first introduce constructivist methods for artificial learning and the issues related to real application such as in an ambient intelligent system. In this article we focus on the problem of bootstrapping the learning of sensori-motor regularities starting from raw signals, which is a first step in the more general model of a constructivist learning. Then we will present our approach to this problem, based on a decentralized approach using a self-organizing multi-agents system, and provide some preliminary results.

Research issues and related works

Before going any further, let us consider an illustrative example of the addressed research problem. Unknown sensors $\{s_1; s_2...s_n\}$ and actuators $\{a_1; a_2...a_n\}$ are connected to an ambient intelligent system S. S is considered as an autonomous agent that must be able to learn the effects of, for instance, a_x on s_y and s_z . If S receives an objective from the user that involves one of these perceptions/actions, it should be able to take the knowledge of those interactions into account. For example, let us say that a_x is a wireless actionable plug with an unknown device D connected on it. If D is an electric heating, S should become aware that it can affect both temperature (s_y) and energy consumption (s_z) by switching a_x , whereas it can affect luminosity and energy consumption if D is a lamp. These are simple regularities that may seem trivial to learn. This is true if the system's designer defines an a priori representation, for example the definition of events and of how to manage time relations. By doing so, a learning agent may directly focus on what is relevant, and find attended regularities out of a reduced search space. Here we consider the case where the agent has no a priori representation and must both discover how to grasp raw signals and how to learn from them at the same time.

Continuous environment and the discretization process

In real world environments, if we want to avoid an a priori definition of a model, the system has to deal with the complexity of raw data coming continuously from sensors. The perceptual aliasing problem is one of the main issues: the learning agent has to learn precisely what are the boundaries of events both in time and space. But the task of separating an event out of the thread of experience is quite difficult. A human brain seems to be able to focus on the relevant details of experience in order to identify a situation and ignore anything else. In classical AI, particularly with logic based symbolic representation, the exhaustive characterization of a situation is impossible in large environments (Frame Problem), most importantly when considering a large buffer memory of past events. More generally, the question of dealing with the complexity of the real world without using a model or symbolic representation is also known as the Moravec's paradox. This paradox emphasizes the fact that basic sensorimotor skills of living organisms are more difficult to reproduce artificially than high level cognitive abilities, as argued by Brooks (1991). Handling continuous experience is a particularly significant issue for constructivist learning methods, related to their application to real world problems.

Developmental robotics

The field of developmental robotics proposes interesting works in accordance with the constructivist principles presented above and is confronted with the problem of handling continuous environments (see Lungarella et al. (2003) for a review). This research field stems from the assessment that biological organisms go through a period of development before reaching their final form. One of the main ideas is that learning must be facilitated by the progressive increase of the complexity of the environment perceived by the agent while progressively acquiring highly tuned skills. Indeed, at the beginning of his growth, an infant has limited sensor and motor abilities, as well as limitation in his nervous system. At first sight this may be considered as a defect, but we can also consider that it is an advantage regarding the learning process, since it enables to decrease or prevent any data overload, that would not be useful for the first development steps. If the agent is precociously facing a huge quantity of information, learning can be intractable. Thus the management of the development of an artificial agent during the learning process could be an improvement. In developmental robotics, the programmer does not specify a

targeted task; the robot must freely explore its environment by interaction. For example, an *intrinsic motivation* could be expressed in the robot by a certain curiosity which leads it to focus on interesting things, like situations which are neither too predictable nor not enough, as proposed for example by Oudeyer et al. (2007). The capacity of making predictions is a part of the mechanism of development, from which the cognition is built. Cognitive functions of higher level, such as planning, may be seen as a prediction of series of events based on experience. Thus, in continuous environments the definition of the notion of event is crucial. Without pre-given representation, the agent has to learn what an event is, and at the same time it uses events to learn patterns. Before presenting a possible application of such a developmental approach to an ambient intelligent system, it is worth introducing more precisely the constructivist learning paradigm in AI and works related to the problem of bootstrapping regularities from raw experience.

Constructivist learning

In psychology the constructivist theory especially developed by Piaget (1954) postulates that a learning agent constructs its own representation of its world, which it uses to give a meaning to its experience (assimilation). Learning, therefore, is simply the process of adjusting its mental models to fit in new experiences (accommodation). Following Drescher (1991) who takes inspiration from the work of Piaget to propose a model of constructivist learning, many researchers work on the schema learning technique. In this model the pattern of interaction used to build the representation of the agent is called a schema. It consists mainly in three elements: a context, an action and a result. The meaning of a schema is: if the agent executes the action A in the context C, it predicts the result R. Schemas are learned incrementally while the agent performs random actions, in the manner of an infant fumbling. A schema is an interesting structure for a representation since it relates an action to its consequences in an environment, based on agents experience. The learning is not goal-oriented, contrary to classical reinforcement learning for example, where the agent learns which action to execute in which context, depending on the purpose of an external learning task. Here, the agent predicts the consequences of its action, and this knowledge may be used for different purposes. The learning model of Drescher (1991) is very interesting because it offers to agents the possibility to learn abstract concepts grounded in experience thanks to the creation of synthetic items (see Perroto et al. (2010)). However, this model operates on a discrete environment and is quite resource consuming, so that it cannot be applied directly to continuous environments. Also many other research in constructivist learning methods in AI provide valuable results but are often designed for simplified simulated environments (see Guerin (2011) for a survey).

Related works

There are works that propose solutions to overcome the problem of handling continuous environments. For example we can mention Chaput et al. (2003) who advises to use a hierarchy of self-organizing maps (SOM) to improve the schema mechanism of Drescher. In their experiment the environment is represented by binary vectors and the time is managed in a step-based manner, but they argue that this improvement may enable constructivist learning for realistic environments. Further work of Provost et al. (2006) steps in this direction with a system that couples a SOM based discretization process of environment with reinforcement learning for a robot navigation problem.

Linaker and Jacobsson (2001) propose a system that features an event extraction mechanism (classification system) which converts raw multi-dimensional and time-step based sensor data into series of events of a larger time scale. This discretization of raw data into events enables to learn a policy of action on a higher level with a reinforcement learning mechanism. It is clear that the low level event extraction mechanism will drastically influence the other parts of the learning system. The major problem for such an approach lies in the fact that the classification process is unique and defined a priori. Yet, the possible classes generated by the discretization mechanism may differ according to the different patterns in which the variables are involved. A solution to this problem is to let the discretization process evolve, guided by a feedback from the higher levels of learning, so that the classes are always adapted to the learned pattern.

This is exactly the case in the model of Mugan and Kuipers (2007) who propose an event extraction mechanism coupled with a learning system of predictive rules. Events are extracted from the continuous experience as transitions between states that are not predefined. Then predictive rules are learned in order to associate an event to an other. An interesting aspect of this system comes from the feedback provided by the learning system to the event extraction mechanism that enables to create new states, and thus learn refined correlations. Thus, the agent learns to perceive in a way that enables to learn more efficiently. This work enhances the developmental process that interconnects the discretization of experience with learning. However, each aspects of this process (the discretization algorithm, the predictive rules,...) are particular implementations that suit with the specific problem considered. In the following of this paper, our purpose is to make explicit the fundamental components of such a process, and propose a generic model that allows multiple implementations of each aspect, motivated by handling heterogeneous unknown environments.

Model presentation

Learning following a constructivist approach is a promising trail in order to build autonomous agents strongly adapted to their environment. In particular, since there is no assumption

about the representation and the objectives of the system, it suits well the problem where both the agent's sensori-motor abilities and its objectives are initially unspecified. Ambient intelligent systems for example present these characteristics. Like robots, they are considered as autonomous systems composed of sensors and actuators. Particularly for AmI systems, constructivist learning is a possible solution to face the fact that sensori-motor abilities are unknown and evolve over time (a user may add, move or remove sensors and actuators at anytime). Moreover, the large variety of equiment results in countless combinations that make harder the design of an expert representation. In the previous part we mentioned the difficulty to handle continuous environments for constructivist learning methods, in this section we propose a solution to deal with this problem, with the example of an ambient intelligent system.

Description of the problem

Developmental learning relies on agent-environment interactions, that is to say on signals exchanged at the frontier between the agent and its environment. In this model this frontier is represented by a set of numeric continuous variables which represents sensors and actuators $V = \{v_1; v_2; ...; v_n\}$. As illustrated on Figure 1 (a), these variables evolve in time according to their respective allowed values. Let us denote $v_i(t)$ the value of v_i at time t. This continuous flow of signals is interpreted by the agent as an event sequence. In the following we call $experience\ E = \{e_1; e_2; ...; e_n\}$ the set of events constructed by the agent. We assume that there are some regularities in this experience. A regularity can be represented as a pattern that the agent is able to isolate within its experiential space and identify thanks to its recurring nature.

From the point of view of the learning system, there is no difference between variables connected to actuators and variables connected to sensors. So the patterns learned by the system might indifferently involve actions and perceptions. However, it is clear that some actions have to be performed in order to appear in some patterns. Contrary to robotics, the domain of ambient intelligence does not really allow us to let the agent randomly explore its possible interactions with its environment in the first place. A motor babbling behavior (see Mugan and Kuipers (2007)) might be lengthy and its consequences would definitely constitute a nuisance for users, for obvious reasons. Hopefully, an ambient intelligence system presents an interesting property that we can exploit to face this problem: a user is able to directly perform actions. For safety and ethical reasons, a user must always keep control of its environment. So as a design constraint, we consider that for every action that can automatically be performed by the agent, a user should have the possibility to directly perform it. We make the hypothesis that the agent is always able to perceive such an action performed by a user, as a kind of *proprioception*¹, so that if this action

¹the sense of internally perceiving our own movements

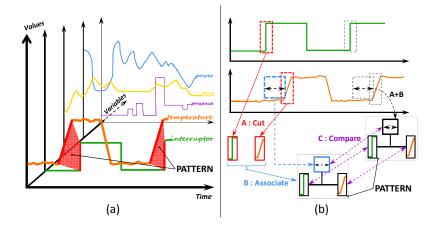


Figure 1: (a) The continuous experience of the system expressed as a set of variables.
(b) The 3 types of operations that enable the construction of patterns. A: Find a temporal reference in order to perceive a variation. B: Find associations between variations. C: In order to reinforce existing structures, modify them or create new ones, it is necessary to be able to compare elements of representation.

becomes involved in a pattern, the agent would be able to perform it by itself later. Najjar and Reignier (2013) refer to this as an *observed action*. Thus the initial random motor babbling is replaced by actions driven by the user, which get involved at the very beginning of the learning process as a form of *motivation* for the developing agent.

Constructing patterns and the bootstrapping problem

A pattern is relevant if it contributes to the creation of new patterns, so that to enable the construction of a representation which justifies them. In other words, the agent has to learn how to perceive efficiently in order to be able to learn more. This circular definition raises a bootstrapping problem similar to the one mentioned by Drescher (1991): finding a context to characterize the association between an action and a result requires to know formerly this association, but learning this association is difficult without knowing the context. This kind of *teleological* issue is typical of the study of living systems.

The purpose of the elementary structure of representation is to express the primary kind of regularity that the agent is able to learn in the first place. For example *schemas* (Drescher (1991)) express a logical and temporal relation between three elements. The *result* is likely to be experienced after the *action* if the *context* was experienced initially. But starting with a set of continuous variables as experience, this raises many questions. What are precisely a context, an action and a result? What are the duration of these elements and their relationships? It seems that these notions are already relative to a form of representation. Of course this issue is avoided if there is an a priori abstraction performed by the designer of the system in order to discretize the environment and guide the learning process.

On the other hand one may consider that the schema or any similar kind of structure like the *functional circle* from Von Uexküll (1992), constitutes an undividable elementary structure that is intrinsic to the agent. Thus, Georgeon and Aha (2013) define the *sensori-motor interaction* as such a primitive. This is reminiscent of the *subsumption architecture* from Brooks (1991), but such methods require to formerly implement such functional circles, and thus to know precisely the agent and its environment. Moreover, this preliminary hardcoding of elementary behaviors may be laborious in complex fields such as ambient intelligence systems.

Apart from specifying arbitrary initial actions and perceptions, one way to solve this problem is to consider a developmental approach. At the beginning of the learning process the agent is fitted with rough sensori-motor abilities which enable it to learn primary patterns, then its abilities develop and the learning of a more accurate representation becomes possible. At one point, the already learned representation can guide the development of the abilities. In biological organisms this task is most probably realized both during phylogenesis and ontogenesis, but it needs to be expressed as a single problem for artificial systems. The "bootstrap learning" as presented in Kuipers et al. (2006) corresponds to this critical problem of getting primary patterns from raw data of uninterpreted sensors (see also Guerin (2011) for a review).

In our model, the elementary structure of representation is composed of two generic notions of *events*. First, a *perception* event is a selected moment in the evolution of a variable, for example a variation or a stable state. Second, an *association* event describes a relation between two events, for example a duration between the occurrences of two perceptions. So it is possible to express a *schema-like* regularity with a combination of association and perception events.

More generally, events involved in associations may be associations, therefore enabling the construction of more complex patterns.

A multi-agent system for the construction of patterns

Our proposal is based on a self-organizing multi-agent system (MAS), where three populations of agents² will interact with each other, guided by a feedback from the global system activity in order to construct relevant patterns and provide a grounded representation. The multi-agent architecture suits naturally the view considering the construction of representation as a self-organization process, as well as the heterogeneity of the domain, and the necessity for the system to be resilient. In the MAS, agents perform a function corresponding to their role in the system, that they can change to explore the possibilities to construct a representation. Rather than specifying an a priori model of perception, or using an exhaustive search, this model relies on the elaboration of collaboration, reinforcement, and internal dynamics within the system. Related to the generic notions of perception and association, we identify three primary functions that are necessary to bootstrap the learning of elementary patterns from the continuous experience as illustrated in Figure 1 (b).

• The first operation "cut" corresponds to the creation of a perception event. The set of possible cut functions is $F^c = \{f_1^c; f_2^c; ...; f_n^c\}$. A cut function that transforms values into events is defined as

$$f_i^c: v_i \to \{e_1^c; e_2^c; ...; e_n^c\}$$
$$V \to E^c$$

with E^c the set of perception events.

In the multi-agent system, *cut* functions are implemented by *Perception agents* which are connected to a continuous variable. These agents create events that represent a particular time period of the evolution of the variable. For example, a *Temporal Window Agent* may produce perceptions based on sliding windows of a given duration. In this case a parameter that conditions the behavior of the agent could be the duration of the window. If the parameter evolves due to feedback received by the agent, it corresponds to the exploration of a new *cut* function.

• The second function type, *compare* functions $(F^s = \{f_1^s; f_2^s; ...; f_n^s\})$ which serve to evaluate the similarity between two events, is necessary to classify and manipulate

events. A *similarity* function compares events to determine if they are similar, and is defined as

$$f_i^s: e_x, e_y \to s$$

 $E \to [0, 1]$

where e_x, e_y are created by the same function (s=1 means totally similar, s=0 totally different).

On Figure 2 we present an example of the activity between a *Perception agent* and a *Similarity agent*. A threshold is defined for s in order to decide the similarity and enable classification (e.g. 0.9). For example, the *temporal window agent* creates an event ec1 which contains the variations compared to the average value of the sliding window. Later another event ec2 is created by this agent and may be considered as similar or different depending on the behavior of the similarity agent involved. If the similarity agent implements a general function f_1^s (intervals of size 3), the two events look similar (3 occurrences in each interval), whereas if it implements a more specific function f_3^s (intervals of size 1) the events are perceived as different.

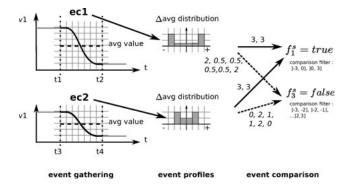


Figure 2: Interaction between a perception agent and a similarity agent.

• Finally the last functions are associate functions $(F^a = \{f_1^a; f_2^a; ...; f_n^a\})$ leading to the creation of association events. An association function that creates an event from two other events is defined as

$$f_i^a: e_x, e_y \to e_z^a$$

 $E \to E^a$

 $(E = E^c \cup E^a)$. For example a *Binary Duration Agent* is defined for two *source events* e_1 and e_2 , and creates an association event for every occurrence of e_1 , characterized by the duration with the previous occurrence of e_2 .

A *Similarity Agent* is associated with a perception or an association agent, in order to classify events created by these agents. For instance, on Figure 3, events are created from

²In the following we will refer to the above-mentioned autonomous learning agent as "the system" in order to avoid any ambiguity with the MAS's agents.

the activity of variables v_1 and v_2 by agents which implement functions f_1^c and $f_{1'}^c$. Different occurrences of e_1^c and $e_{1'}^c$ (source events) are classified by similarity agents. An association agent creates an event e_1^a based on the occurrences of events e_1^c and $e_{1'}^c$, and then creates another event e_2^a later when new occurrences e_2^c and $e_{2'}^c$, are perceived as similar to e_1^c and $e_{1'}^c$. Another similarity agent implementing a function f_2^s compares the association events generated by the association agent f_1^a in order to classify them (as presented on Figure 5).

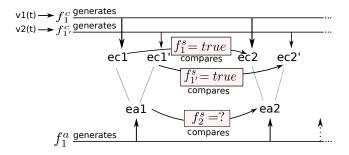


Figure 3: Interactions between the three types of agent.

The evaluation of representation and feedback process

The search space of the system is the set of possible functions $F = F^C \cup C^A \cup F^S$. The relevance of a function can be evaluated according to its participation to the construction of patterns, which is necessarily performed in collaboration with other functions' types. We can represent this feedback by a fitness function $Q: f \in F \to [0,1]$ that defines a function's quality. Then the goal of the system is to find a combination of functions that enables to construct relevant patterns. In other words select the optimal set of functions that maximize the overall quality evaluation. This can be defined as $F' \subset F = \{f' \in F' | Q(f') > T_Q\}$ where T_Q is a threshold for the fitness function.

As illustrated on Figure 4, the mechanism is a circular process initiated by the creation of representation elements from the experience discretized by the *cut* operations. Each time a new element is created, it is compared with previously created elements coming from the same source, and it is fused with a similar existing element or constitutes a new distinction, to form classes of events. Then events are used to create new events with the associate operations, which in turn are compared and so on. These generic functions and the associated structures enable to model a basis for the mechanism of extracting sensori-motor patterns, but there are many ways to instantiate each of them. A given version of a function may suit well the learning of a specific aspect of the system's experience at a certain level, and fail to describe another one. So in this model these different possibilities are processed in parallel, and the best ones are reinforced because they enable the construction of a stable representation. Each instance of a function (an agent) has to evaluate if it contributes to a viable representation or not.

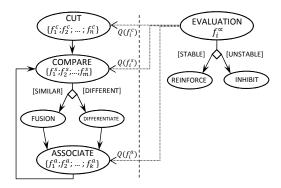


Figure 4: Process of bootstrapping pattern construction according to three types of functions.

The evaluation of patterns may correspond to their ability to make reliable predictions, as used by Mugan and Kuipers (2007). Indeed the adequacy of the representation is tied to the notion of prediction as stated by Von Glasersfeld (1984):

"Quite generally, our knowledge is useful, relevant, viable, or however we want to call the positive end of the scale of evaluation, if it stands up to experience and enables us to make predictions and to bring about or avoid, as the case may be, certain phenomena (i.e., appearances, events, experiences). If knowledge does not serve that purpose, it becomes questionable, unreliable, useless, and is eventually devaluated as superstition."

But in the case of the bootstrapping problem, it is not possible to evaluate the prediction abilities of patterns until we have patterns to consider as a predictive structure. Thus, we introduce a transitional evaluation of *interest*, in order to drive agents to promising areas of the search space and use predictions as a more important feedback in a second step. The interest of primary events is defined according to the notions of specificity and stability. If functions are much too accurate, the classes of events are likely to multiply without being reinforced, and will not constitute regularities. On the contrary, if the operations are much too general, the representation might be quite stable but also unexpressive. Therefore, in order to give feedback to agents, the representation must be evaluated in terms of a tradeoff between accuracy and stability. This intuitive idea is used in order to overcome the bootstrapping problem. Indeed, it enables to guide the activity of the initial rough functions, before first patterns are formed. The specificity of an event can be expressed in relation with a reference event, which corresponds to the more general class, that is to say all occurences of the event. For example for a Binary duration agent, the reference is a set of durations between random events. The *specificity* of an event is: $s(e) = 1 - f^s(e, ref)$ defined between 0 and 1. Let us note |e| the number of occurences of the event $e \in E^{\alpha}$, and E^{α} is the set of event classes generated by the similarity agent among all occurences. The *weight* of an event is defined according to the repartition of the occurences of this event in relation with the other specific events:

$$w(e) = \frac{|e|}{\sum\limits_{s(e_i) > \epsilon; e_i \in E^{\alpha}} |e_i|}$$

The interest of an event is i(e) = min(s(e), w(e)). As illustrated on Figure 5, an association event which has both a high specificity value and weight value, may represent a regularity of the duration between two events, and give a positive feedback to agents that construct these events.

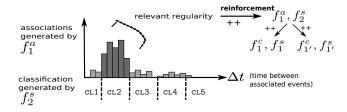


Figure 5: Feedback from an association to participating agents.

The purpose of this architecture is to investigate the dynamics of bootstrapping regularities from continuous signals as a developmental process, without committing to a specific implementation. This is the first step in a more general model of constructivist learning. We let the opportunity to incorporate a variety of agents at different levels into the system for each of the operations involved, in line with the ideas developed by Minsky (1991). In the last section we present preliminary results focusing on the bootstrapping step with a system composed of very simple agents in order to experiment the global mechanism.

Results and conclusion

The experimental setup consists of a simple set of three interacting numeric variables V_1 ; V_2 ; V_3 . V_1 is an interruptor that actions V_2 and V_3 , which vary at different paces and scales. Activations of V_1 are repeated randomly between 2 and 6 seconds after the end of the series of effects on other variables, as represented on Figure 6. The purpose is to verify that structures generated by the system can stabilize on relevant associations, which can enable to bootstrap the pattern learning.

In this experiment, the system is initially composed of a minimal number of elementary agents: one perception agent per variable (a temporal window agent mentioned above which collects variations in a sliding window), and one similarity agent. Each event is represented by a vector (histogram), storing the distribution of the collected values.

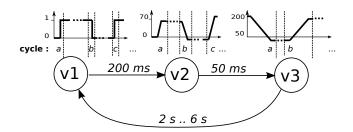


Figure 6: Experimental setup.

Thus the similarity agent compares the intersection percentage between two vectors. At the beginning there is no association agent, since there is no event to connect with. Events that go beyond a certain threshold of interest lead to the creation of an association agent, but only if the pair of agents (perception agent & similarity agent) is stable. A mesure of stability takes into account the changes of important criteria for the pair, such as the number of event classes and the respective interest of each events. Stable pairs that do not include any interesting event arouse a negative feedback on their agents, and lead to the creation of new agents for exploration and encourage agents to use other functions.

So when an association agent is connected to two events that match with a regularity, it must appear as an interesting event. For example Figure 7 shows the first interesting association event that is found (after 34 activations of V1). This association forms a three-events pattern which corresponds to the regularity involving the variation of V1 followed by the variation of V2.

As expected this kind of rough pattern can enable to bootstrap a more elaborated learning mechanism since it contains necessary information about the events and their relation in time to construct a predictive structure. In a second step, there can be a feedback from the prediction mechanism to the discretization process, so that event perception learning is oriented according to its utility for predictions. Further work will be mainly dedicated to the implementation of this mechanism, but also includes: investigate the application of this generic process to higher level in the representation; and also study the possible internal dynamics within the population of agents that can be exploited to improve the performance of the system, and develop self-organization process. In this article, we point out the fundamental problem of bootstrapping learning of sensori-motor patterns from continuous environments in a constructivist perspective of learning. We propose to tackle this problem with an unconventional decentralized approach. The multi-agent architecture is used to model the self-organization process of the construction of representation and the dynamic between learning and the discretization of experience. This model is still in an early stage of development but presents interesting theoric and applicative perspectives. We show that the proposed

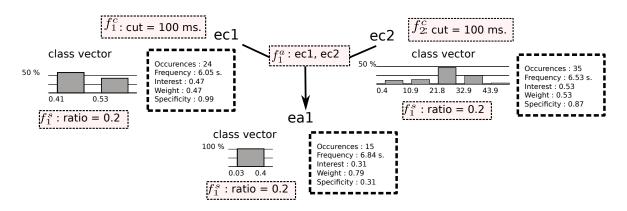


Figure 7: Example of a pattern.

mechanisms lead to the recognition of significative events and regularities without any a priori knowledge. Despite its apparent simplicity, the problem we address is complex if not avoided thanks to an abstraction effort in the design of a preliminary representation. These results are very encouraging and reveal a great potential for such an approach. Going deeper in the understanding of such mechanisms will allow to efficiently address many problems. Applicative fields such as Ambient Intelligence features both learning issues related to the undeterminism of the system, and the practical constraint to handle continuous environments.

References

- Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence*, 47(13):139 159.
- Chaput, H. H., Kuipers, B., and Miikkulainen, R. (2003). Constructivist learning: A neural implementation of the schema mechanism. In *In Proceedings of the Workshop on SelfOrganizing Maps (WSOM03.*
- Drescher, G. (1991). *Made-up minds: a constructivist approach to artificial intelligence*. The MIT Press.
- Georgeon, O. and Aha, D. (2013). The Radical Interactionism Conceptual Commitment. *Journal of Artificial General Intelligence*, 4(2):31–36.
- Guerin, F. (2011). Learning like a baby: a survey of artificial intelligence approaches. *The Knowledge Engineering Review*, 26:209–236.
- Kuipers, B. J., Beeson, P., Modayil, J., and Provost, J. (2006). Bootstrap learning of foundational representations. *Connection Science*, 18(2):145–158.
- Linaker, F. and Jacobsson, H. (2001). Mobile robot learning of delayed response tasks through event extraction: A solution to the road sign problem and beyond. In *International Joint* conference on artificial intelligence, volume 17, pages 777– 782. Lawrence erlbaum associates ltd.
- Lungarella, M., Metta, G., Pfeifer, R., and Sandini, G. (2003).

 Developmental robotics: a survey. *Connection Science*, 15(4):151–190.

- Minsky, M. (1991). Logical versus analogical or symbolic versus connectionist or neat versus scruffy. *AI magazine*, 12(2):34.
- Mugan, J. and Kuipers, B. (2007). Learning distinctions and rules in a continuous world through active exploration. In *Proceedings of the Seventh International Conference on Epigenetic Robotics (EpiRob-07)*, pages 101–108.
- Najjar, A. and Reignier, P. (2013). Constructivist Ambient Intelligent Agent for Smart Environments. In PerCom IEEE International Conference on Pervasive Computing and Communications, San Diego, États-Unis.
- Oudeyer, P.-Y., Kaplan, F., and Hafner, V. (2007). Intrinsic motivation systems for autonomous mental development. Evolutionary Computation, IEEE Transactions on, 11(2):265–286.
- Perroto, F., Alvarez, I., and Buisson, J.-C. (2010). Constructivist Anticipatory Learning Mechanism (CALM): Dealing with Partially Deterministic and Partially Observable Environments. In *International Conference on Epigenetic Robotics* (*EpiRob*), pages 110–120. Lund University Cognitive Science.
- Piaget, J. (1954). *The Construction of Reality in the Child*. Basic Books.
- Provost, J., Kuipers, B. J., and Miikkulainen, R. (2006). Developing navigation behavior through self-organizing distinctive-state abstraction. *Connection Science*, 18(2):159–172.
- Riegler, A. (2001). Towards a radical constructivist understanding of science. *Foundations of Science*, 6(1):1–30.
- Varela, F. J., Thompson, E., and Rosch, E. (1991). The embodied mind: Cognitive science and human experience. The MIT Press.
- Von Glasersfeld, E. (1984). An introduction to radical constructivism. *The invented reality*, pages 17–40.
- Von Uexküll, J. (1992). A stroll through the worlds of animals and men: A picture book of invisible worlds. *Semiotica*, 89(4):319–391.
- Ziemke, T. (2001). The construction of reality in the robot: Constructivist perspectives on situated artificial intelligence and adaptive robotics. *Foundations of Science*, 6(1-3):163–233.

Habit-based Regulation of Essential Variables

Matthew Egbert¹ and Lola Cañamero¹

¹ Embodied Emotion, Cognition and (Inter-)Action Lab, University of Hertfordshire, School of Computer Science, Hatfield, UK, AL10 9AB mde@matthewegbert.com

Abstract

A variety of models have been developed to investigate "homeostatic adaptation," a mechanism inspired by Ashby's homeostat, where a plastic control medium is reorganized until one or more essential variables are maintained within predefined limits. In these models, "habits" emerge, defined as behavior-generating mechanisms that rely upon their own influence to maintain the conditions necessary for their own persistence. In this paper, we present a recently developed sensorimotor-habit-based controller that is coupled to a simulated two-wheeled robot with a simulated metabolism. The simulation is used to demonstrate how habits can have the same essential variable(s) as the metabolic or "biological" organism that is performing the behavior, and that in certain conditions when this is the case, the emergent habits will tend to stabilize essential variables within viability limits. The model also demonstrates that an explicit pre-specification of (A) which variables should induce plasticity and (B) which values of those variables should induce plasticity is not always necessary for homeostatic adaptation of behavior.

Introduction

Over the past few decades, a range of "embodied" approaches to the study of the mind have emerged, including: enaction (Stewart et al., 2010), dynamical approaches to cognition (Beer, 1997), the sensorimotor approach to perception (O'Regan and Noë, 2001), and embodied AI/Robotics (Brooks, 1990; Pfeifer and Bongard, 2006). Unifying these approaches is the notion that instead of investigating abstract computational problems such as chess, research in robotics and AI should instead focus upon embodied, embedded agents that use ongoing interactions with their environment to accomplish "intelligent behavior," where intelligence is evaluated by its contribution to the survival of the agent, (see e.g. Steels, 1995). But, the notion of system survival or "viability" in the context of robotics can be more challenging than it first appears. Biological organisms are far-from-equilibrium, dissipative structures that depend upon ongoing processes of self-production and selfmaintenance to counteract their tendency to degrade (Kauffman, 2000; Nicolis and Prigogine, 1977; Schrödinger, 1944; Maturana and Varela, 1980), but conventional robots do not share this property and thus there can be challenges in relating the behavior of artifacts to the behavior of biological organisms. This difficulty has led to different emphasis placed upon the metabolic organization within the embodiment community, with some proponents more or less ignoring the concept and others arguing that the metabolic organization is fundamental (Froese and Ziemke, 2009). As an example of the former, metabolism is seen as largely irrelevant to the sensorimotor approach to perception, with suggestions that a missile guidance system could have some minimal perceptions (O'Regan and Noë, 2001, p. 82) and that a machine, if sufficiently complex, could be considered aware (with no mention of metabolism / self-production). In contrast, many studies in embodied- and evolutionary-robotics acknowledge the metabolic-organization and its role of defining viability limits as important and set out to investigate how regulation can be accomplished that maintains viability. Most of these, however, do not model in detail the processes through which the viability-limits are defined (e.g. metabolism), as the focus is instead upon how a particular form of regulation can be accomplished (e.g. Avila-Garcia and Cañamero, 2004; Beer, 1995), but some of our recent models include in detail both behavioral dynamics and the metabolic dynamics that define the viability limits. Using these models, we have demonstrated how organisms can respond to the processes that define their viability limits to perform an ongoing, in-the-moment evaluation of the environment (Egbert et al., 2009), that allows an organism to behave in a history-dependent and adaptive way that integrates multiple environmental and internal factors into an adaptive behavior (Egbert et al., 2010b; Egbert, 2013) that may play an interesting role of facilitating adaptive evolution (Egbert et al., 2010a).

Some "enactivists" (Stewart et al., 2010) believe that the metabolic-organization fundamentally underlies mind. In particular, it has been argued that the metabolic-organization underlies a modern concept of intrinsic teleology, in which an organism's survival grounds teleological (Weber and Varela, 2002) and normative (Barandiaran and Egbert, 2013) descriptions of its behavior. In this view, organisms act to

satisfy the needs that are the result of their own precarious, metabolic organization (Weber and Varela, 2002). But of course, not all behavior, adaptive or otherwise, is a direct response to biological needs and dynamics; not all behavior occurs to maintain biological essential variables within viability limits, and so what can biological embodiment contribute, to "higher" forms of adaptive behavior, i. e. beyond the metabolism-based chemotaxis of bacteria? One proposal stems from the observation that aspects of perception, learning and sensorimotor behavior appear to have properties in common with self-maintaining organisms. In particular, an analogy is drawn between habits seen as self-maintaining patterns of behavior, and biological organisms seen as selfmaintaining metabolic entities (Di Paolo, 2003; Barandiaran, 2007, 2008). By viewing habits as having the same precarious, self-maintaining organization as biological entities, it becomes possible to understand how, in a way similar to the grounding of many survival-based behaviors in metabolism, other behaviors can be grounded in the needs of habits themselves. In metabolism-based chemotaxis, bacteria act to maintain their biological essential variables within limits and analogously, habit-based behavior maintains the conditions necessary for the persistence of the habit itself, i.e. the persistence of the behaviour and the mechanism(s) that generate it. Put another way, biological embodiment grounds survival, the "mother of all values" (Weber and Varela, 2002; Di Paolo, 2003), and habits are autonomous "mental life forms," that ground the others (Di Paolo, 2003; Barandiaran, 2007, 2008).

In a set of minimalistic computational models, Di Paolo (2000, 2003) demonstrated how a behavior-generating mechanism can stabilize the conditions necessary for its own persistence. The first model involved a simple simulated two-wheel, two-light-sensor robot, controlled by a plastic continuous time recurrent neural network. In a design inspired by Ashby's homeostat (Ashby, 1952), the weights of the connections in the neural network were fixed when the neural firing-rate was within a predefined region, but outside of this region the weights changed according to deterministic plasticity rules. A genetic algorithm succeeded at identifying model parameters (initial weights, which plasticity rules applied to which synapses, etc.), that resulted in networks that both avoid plasticity and perform phototaxis, and we can say that the neural network configurations that emerged in this model were self-maintaining behaviorgenerating mechanisms. Why? A configuration of weights in the neural network is precarious in the sense that it can only persist if plasticity is avoided and it has viability-limits in that neuron firing rates must not leave predefined bounds if the behavior-generating mechanism is to persist. The firing rates of the neurons are influenced by internal neuron activity, but also by the sensorimotor dynamics of the behavior: the neurons influence the motors which influence how the environment influences the sensors, which, coming

full circle, influence the neurons. The stability of a particular configuration of weights of the NN therefore depends, in part, upon the behavior that it drives, and thus can be considered to be a self-maintaining behavior-generating mechanism.

In this first model of Di Paolo's, there is no modeled metabolism, biological essential variables or biological viability limits. The behavior is phototaxis because of selection pressures applied during the artificial evolution. We could imagine that the agent has a biological (i. e. metabolic) need to move toward the light, but even so, the coupling of this need and the self-maintaining behavioral mechanism was accomplished through an evolutionary process. Is there a way that the self-maintaining behavior-generating mechanisms (henceforth "habits") and the self-maintenance of the organism could be more tightly integrated? In order to address this, Di Paolo (2003) developed a second model consisting of a simulated, Braitenberg-inspired robot, with plastic mappings between sensors and motors. These mappings undergo stochastic change when the battery (an analog to metabolism) is outside of certain pre-defined limits. This is a step forward in that the biological needs and the habits are intertwined in a cycle of dependence, where the maintenance of a viable battery level depends upon the habit, but also the stability of the habit depends upon the battery level remaining within bounds, for if the habit drives the metabolism outside of the biological viability limits, the behavior-generating mechanism will also lose viability due to the stochastic weight-change process. In a different approach to coupling behavior to metabolic needs, Iizuka et al. (2013) selected the limits for the induction of plasticity such that when there was no sensor activity, the weights of the network would change. In this way, the system could only become stable when it was performing phototaxis. The relationship between the essential variable (a "photosynthetic metabolism" that justifies phototaxis as a target behavior) and the behavior has been hard-wired by the designer, in the sense that the robot can adapt to inversions of its visual field, but could not adapt to its phototactic-needs becoming photophobic-needs.

For all of the models presented above, the limits of plasticity, i. e. the surface between the states where plasticity does occur and where it does not, are parameters of the model that are pre-specified. It is also the case that for every model, one or more variables are given a special privileged status as a "plasticity inducing variable". In the first model of Di Paolo's presented above, and in Iizuka's model, these are the neuron firing rates. In the second model of Di Paolo's, it is the state of the "biological" essential variable, i.e. the battery state, and it is through this explicit specialization of this variable that the biological dynamics and the mental dynamics are coupled. In this paper, we use a newly developed model of habits, coupled to a simulated two-wheeled robot with simulated metabolic dynam-

ics to demonstrate how it can be possible to couple selfmaintaining behavior-generating mechanisms with biological essential variables without pre-specifying the region of plasticity and moreover, without pre-specifying which variables are plasticity-inducing variables.

In the next section we describe the robot, its simulated metabolism and its habit-based controller. We then describe two scenarios that we use to demonstrate that by including an interoceptive sensitivty to the state of the metabolism as a sensory-variable, self-reinforcing patterns of behavior emerge that stabilize that metabolic dynamics. We present the results of simulations of these scenarios before interpreting and discussing them.

Simulation

Robot and environment. We simulate an simple robotic agent embedded in a two-dimensional square environment 8 units wide, with periodic boundary conditions. The robot has two directional light sensors and two independently driven motorized wheels. The motion of the robot is $\dot{x} =$ $\cos(\alpha)(m_l+m_r); \dot{y}=\sin(\alpha)(m_l+m_r); \dot{\alpha}=2(m_r-m_l),$ where x,y is the robots spatial position, $\alpha \in [-\pi,\pi]$ is the robots orientation and $m_l \in [-1,1]$ and $m_r \in [-1,1]$ are the robots left and right motor speeds. The robot's light sensors are located at $x + r \cdot \cos(\alpha + \beta), y + r \cdot \sin(\alpha + \beta)$, where r=0.25 is the robot's radius and $\beta=\pm\pi/3$ is the angular offset of the sensors from α , the heading of the robot. The activation of each sensor is determined by the equation below, where $b = [\cos(\alpha + \beta), \sin(\alpha + \beta)]$ is a unit vector indicating the direction that the sensor is facing, c is the vector from the sensor to the light (which is located in the center of the arena at (0,0), and D is the distance from the sensor to the light.

$$s = \frac{(\boldsymbol{b} \cdot ||\boldsymbol{c}||)^+}{1 + D^2} \tag{1}$$

Metabolism. We simulate the robot as having a metabolism with intrinsic dynamics that are indirectly influenced by the motor behavior. The simulated metabolism is inspired by the blood-sugar dynamics of a diabetic, where hormonal regulation prevents the blood-sugar from diverging, but is insufficient to prevent blood-glucose levels from leaving healthy limits. The dynamics of diabetes has been simulated since this work is part of the ALIZ-E project, where we are investigating how to help diabetic children learn to manage their disease. Part of this support involves developing a better understanding of how behaviors relate to essential variables such as blood-sugar, as well as to reflexive hormonal modulators such as insulin and glucagon (Lewis and Cañamero, 2014), and how diabetes-related (good and bad) habits form and how they can be changed (from bad to good) in support of self-efficacy.

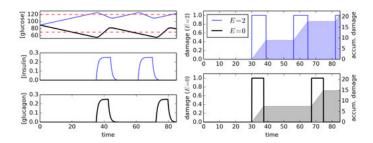


Figure 1: Essential variable dynamics in two worst case scenarios, always feeding (blue) and never feeding (black). Left: Glucose, insulin and glucagon dynamics. Viability limits for the essential variables are indicated by dashed red lines. Right: damage and accumulated damage.

param.	value	description
c	1.0	rate of glucose use / consumption
f_U, f_I	20, 0.1	efficiency of hormonal modulation
b_U,b_I	60.0, 120.0	threshold G-concentrations
c_U, c_I	0.25, 0.25	rate of hormonal production
d_U, d_I	1.0, 1.0	rate of hormone elimination
au	5.0	delay in hormonal response

Table 1: Metabolism-related parameters

The modeled metabolism is not intended to be a realistic simulation of blood-sugar dynamics, but just to qualitatively capture the dynamics of an hormone regulated essential variable that is inadequately regulated, i.e. a variable for which the hormonal regulation is insufficient to keep the system within its viability limits. The model consists of three coupled delayed differential equations, which represent: G, blood-glucose concentration, the essential variable which must remain within limits if the system is to be considered in a healthy state; I, the concentration of insulin, a hormone that removes G from the blood when it is above a threshold, and U, the concentration of glucagon, a hormone that releases G into the blood when it is below a threshold. In these equations, the function $[a < b] \equiv 1$ when a < b, and 0 otherwise.

$$\frac{\mathrm{d}G}{\mathrm{dt}} = E + f_U U - f_I I G - c \tag{2}$$

$$\frac{\mathrm{d}I}{\mathrm{dt}} = [b_I < G_{t-\tau}] c_I - d_I I \tag{3}$$

$$\frac{\mathrm{d}I}{\mathrm{dt}} = [b_I < G_{t-\tau}]c_I - d_I I \tag{3}$$

$$\frac{\mathrm{d}U}{\mathrm{d}t} = [G_{t-\tau} < b_U]c_U - d_U U \tag{4}$$

We defined viability limits such that the system is considered to be healthy if $G \in [b_U, b_I]$. Leaving the viability limits will eventually trigger hormonal regulation of G back into the viability region (with a delay of τ), but the model is configured such that similar to a diabetic, the hormonal regulation is insufficient to maintain G within healthy limits. When the robot is within 2 spatial units of the light it is considered to be "feeding" and the variable E is set to 2, and otherwise E=0. Thus, the behavior of the robot influences G and, as we shall see, if certain patterns of behavior are performed, it is possible for the value of G to remain within the viability limits indefinitely. However, if the robot performs non-ideal behaviors, G will leave the viability limits, and thereby accumulate "damage," defined as the quantity of time when $G \notin [b_U, b_I]$. Figure 1 shows example trajectories for G, I and U for two worst-case behaviors (the robot always eating, and the robot never eating) with the plots on the right indicating the damage and accumulated damage suffered in these scenarios. Throughout this paper, time and time-related values such as τ are specified in arbitrary time-units, where one time-unit is the time in which a robot traveling at full-speed moves 2 spatial units.

Habit-based controller We have recently developed a plastic, self-modifying dynamical system called an Iterant Deformable Sensorimotor Medium (IDSM) (Egbert and Barandiaran, 2014). This system was designed to act as a robot controller that supports the formation of "habits" conceived of as precarious, self-maintaining patterns of sensorimotor behavior. When coupled to a robot's sensors and motors, the IDSM (1) causes the robot to repeat behaviors that it has performed in the past, and (2) allows for the reinforcement of patterns of behavior through repetition, such that the more frequently and recently a pattern of behavior has been performed, the more likely it is to be performed again in the future. If a pattern of behavior is not performed for a period of time, it becomes less likely to be re-enacted, but when behaviors are performed, they become more likely to be repeated in the future, and in this way, self-maintaining patterns of behavior emerge. Metaphorically, the IDSM works similarly to the paths made by animals through the woods or through a field of grass. As sensorimotor trajectories are experienced, pathways are worn in to the IDSM's "sensorimotor-space," such that future sensorimotor pathways are likely to be similar to those pathways that have been taken in the past. In the remainder of this section, we provide an overview of our IDSM architecture. Much of the text here comes from (Egbert and Barandiaran, 2014), which provides a much more detailed description of the IDSM and its dynamics.

The IDSM operates by developing and maintaining a history of sensorimotor (SM) dynamics. This history takes form of many *nodes*, where each node describes the SM-velocity at a SM-state at some point in the past. As the agent behaves, and its SM-state changes, nodes are added, such that a record is constructed of how sensors and motors have changed for various SM-states during the system's history. These are used to determine future motor-actions such that when a familiar SM-state is encountered, the IDSM pro-

Symbol Description

x current SM-state

 N_p SM-state associated with node N (in normalized SM-space coordinates)

 N_v SM-velocity indicated by node N (in normalized SM-space coordinates)

 N_w weight of node N

d(x, y) distance function between two SM-states

 $\omega(N_w)$ function describing how the weight of a node scales its influence

 $\phi(y)$ function describing the "familiarity" (local density of nodes) of SM-state y

Table 2: Glossary of symbols and brief descriptions.

duces behavior that is similar to the behavior that was performed when the agent was in a similar situation in the past.

More formally, each node is a tuple of two vectors and a scalar, $N = \langle \boldsymbol{p}, \boldsymbol{v}, w \rangle$, where \boldsymbol{p} indicates the SM-state associated with the node (referred to as the node's "position" in SM-space), \boldsymbol{v} indicates a velocity of SM-change, and the scalar, w indicates the "weight" of the node, a value that partially determines the overall influence of the node. We shall refer to these components using a subscript notation, where the position, SM-velocity, and weight of node N are written as $N_{\boldsymbol{p}}$ and $N_{\boldsymbol{v}}$ and $N_{\boldsymbol{w}}$, respectively.

As a robot controlled by the IDSM moves through SM-states, new nodes are created recording the SM-velocities experienced at different SM-states. Specifically, when a new node is created, its "position," N_p is set to the current SM-state; its "velocity," N_v is set to the current rate of change in each SM-dimension, and its weight, N_w is set to 0 (an initial value that does not imply that the node is ineffectual, see below). The two vector terms $(N_p$ and N_v) are calculated in a normalized sensorimotor space, where the range of all sensor and motor values are linearly scaled to lie, in each dimension, between 0 and 1.

New nodes are only added when the density of nodes near the current SM-state, as described by the function ϕ , is less than a threshold value, $\phi(\boldsymbol{x}) < k_t = 1$. This density function, ϕ , can be thought of as a measure of how many nodes there are near to the SM-state \boldsymbol{x} , and how heavily weighted those nodes are. Loosely speaking, it is a measure of how "familiar" the SM-state is, and it is calculated by summing a non-linear function of the distance from every node to the current SM-state, $d(N_p, \boldsymbol{x})$, scaled by a sigmoidal function of the node's weight, $\omega(N_w)$, as described in Equations 5–7.

$$\phi(x) = \sum_{N} \omega(N_w) \cdot d(N_p, x)$$
 (5)

$$\omega(N_w) = \frac{2}{1 + \exp(-k_w N_{sv})}; \ k_\omega = 0.025$$
 (6)

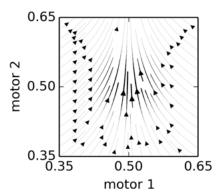


Figure 2: The influence of a single node, with $N_p = (0.5, 0.5)$, $N_v = (0, 0.1)$, and $N_w = 0$. In this didactic scenario, N_v only has a non-zero component in the motor-2 dimension, and thus the node's "velocity" influence causes motor-2 to increase, and its "attraction" influence causes motor-1 to approach a state of 0.5.

$$d(N_{p}, x) = \frac{2}{1 + \exp(k_d ||N_{p} - \boldsymbol{x}||^2)}; \ k_d = 1000 \quad (7)$$

After a node is created, its weight changes according to equation 8, where the first term represents a steady degradation of the node's influence, and the second term represents a strengthening of the node that occurs when the current SM-state is close to the node's position. This latter term allows for the self-reinforcement/self-maintenance of patterns of behavior, such that patterns of behavior that are repeated are more likely to persist than those that are not reinforced.

$$\frac{\mathrm{d}N_w}{\mathrm{d}t} = -1.0 + r(N, \boldsymbol{x}) \tag{8}$$

$$r(N, \boldsymbol{x}) = kd(N_{\boldsymbol{p}}, \boldsymbol{x}) \tag{9}$$

A short period of time after creation (10 simulated time-units), nodes are activated, meaning that they are added to the pool of nodes that influence the motor state. Every activated node influences the motor state, but at any one time only a subset of these will have a substantial influence, for the influence of a node is scaled non-linearly by its distance from the current SM-state by the same distance function used in ϕ above (Equation 7). The influence of each node is also scaled by its weight according to Equation 6, and thus nodes that are close to the current SM-state and nodes with higher weights have a greater influence.

The influence of a node can be broken down into two factors: a "velocity" factor and an "attraction" factor. The velocity factor is simply the motor components of the N_v vector, but the attraction factor, is slightly more complicated. It is a sensorimotor-"force" that draws the system towards the node. This tends to result in a motion in SM-space towards regions of SM-space that are familiar, i.e. for which there is a higher density of nodes and it can compensate for

stochasticity in the environment or perturbations to behavior (see Egbert and Barandiaran, 2014 for details). The attraction vector has its component parallel to N_v removed to prevent it from interfering with the velocity influence of the node (again, see Egbert and Barandiaran, 2014 for details). Figure 2 provides a visualization of the influence of a single node in a hypothetical 2-motor, 0-sensor IDSM. In this example, N_v is exactly vertical, so all horizontal motion is due to the attraction component, and vertical motion is due to the velocity component.

Equations 10-11 describe how the IDSM influences the motor state. The velocity and attraction influences of every node are scaled by the node's weight and distance to the SM-state, and then these are all summed before being scaled by the density of the nodes at the current SM-state such that the influence of all the nodes is averaged and not cumulative. Obviously, the IDSM only has direct control of its motors and the sensor-components of the SM-state are determined by the systems interaction with its environment. Accordingly, the superscripted- μ notation in the equations below indicates where we are only using the motor-components of the indicated vector terms.

$$\frac{\mathrm{d}\boldsymbol{\mu}}{\mathrm{d}t} = \frac{1}{\phi(\boldsymbol{x})} \sum_{N} \omega(N_{\boldsymbol{w}}) \cdot d(N_{\boldsymbol{p}}, \boldsymbol{x}) \cdot (\underbrace{N_{\boldsymbol{v}}}_{\mathrm{Vel.}} + \underbrace{A(N_{\boldsymbol{p}} - \boldsymbol{x}, N_{\boldsymbol{v}})}_{\mathrm{Attraction}})^{\boldsymbol{\mu}}$$

$$A(\boldsymbol{a}, N_{\boldsymbol{v}}) = \boldsymbol{a} - \boldsymbol{a} \cdot \frac{N_{\boldsymbol{v}}}{||N_{\boldsymbol{v}}||}$$

$$(10)$$

Experiment and Control Scenarios We compare two scenarios. In the experimental scenario, the SM-space of the IDSM has two motor dimensions: (the left and right motor of the robot) and three sensory dimensions: its two directional light-sensors, and a direct sensory perception of its essential variable, G (scaled linearly such that the range $G \in [50, 130]$ lies in normalized sensorimotor coordinates in [0,1]). In the control scenario, everything is the same except that the IDSM is not sensitive to G. To keep the total number of SM-dimensions the same in both scenarios, the sensitivity to G was replaced with a motor that has no effect whatsoever. The control is included primarily to show that the task is not trivially solved.

We simulated 25 trials of each scenario. At the start of each trial, we randomly initialized the IDSM with 10000 nodes. These were generated by performing 200 random walks in the 5 dimensional SM-space, each starting from a random location within the SM-space with subsequent loci calculated according to the following equation, $\boldsymbol{l}_{i+1} = \boldsymbol{l}_i + \boldsymbol{r}$, where the components of \boldsymbol{r} are selected from a flat distribution [-0.05, 0.05] and where any components that would take l_i out of the normalized SM-volume are inverted. Nodes were added at each locus of the walk l_i with $N_{\boldsymbol{p}}$ set to l_i , $N_{\boldsymbol{v}}$ set to $l_{i+1} - l_i$, and $N_w = 0$. We then placed

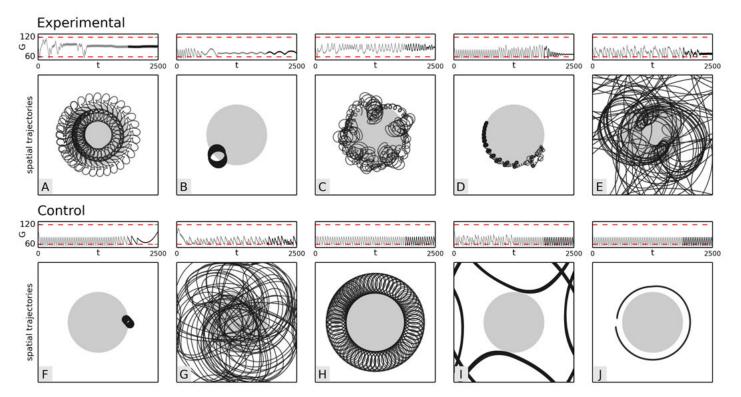


Figure 3: Essential variable and spatial trajectories of the five best performing control and experimental agents. The square plots indicate the spatial trajectories taken by each agent during the final 625 time-units of its simulation (the period during which damage was evaluated). In these plots, the filled-circle indicates the feeding-region. Above each spatial plot, the essential variable, *G* is plotted against time, with the viability-limits indicated by red-dashed lines and the period corresponding to the spatial plot indicated by a darker line.

the robot at a random initial location within the arena, with the essential variable initialized to a value at the center of its viability region (G=90) and the concentration of the regulatory hormones I and U set to 0. We then allowed the IDSM to control the robot and tracked the position of the robot and the trajectories of G,I, and U for 2500 simulated time units, so that we could evaluate the behavior of the robot, and the extent of its success at maintaining the essential variable within viability limits.

Results

To evaluate the performance of each trial, we measured the amount of damage accumulated during the final 625 time-units of each simulation. Figure 3 shows the spatial trajectories and the glucose concentration trajectory plotted against time for the top 5 performing experimental and control trials. We can see that by the end of the simulation, all five of the plotted experimental trials have behavior that maintains G within these limits (although the trajectory in Trial B appears to be on an amplitude-increasing cycle that may eventually leave the viability limits). In comparison, none of the control trials appear to have stabilized G within the viability limits. Three of the experimental agents manage to

avoid incurring any damage during this period, and none of the control agents are as successful. A variety of spatial trajectories can be observed, both in the spatial dynamics and in the dynamics of G. Figure 4 shows the accumulated damage for each trial during this last quarter of the simulation. A Mann-Whitney-Wilcoxon test of these values indicates that the experimental agents are better at maintaining the essential variable within limits to a statistically significant degree (z=-3.13, p<0.002).

Discussion

The IDSM supports the formation of self-maintaining patterns of behavior by (1) assembling a collection of "nodes" that track the SM-state-velocity for different SM-states, (2) using these nodes to drive later behavior, and (3) having these nodes, which perpetually degrade, depend upon a mechanism of self-reinforcement to persist. The self-reinforcement of a node is accomplished by the re-visitation of SM-states near to the node's "position" (N_p) , and so only patterns of behavior that repeatedly visit SM-states can persist. Therefore, in the experimental scenario where the IDSM is sensitive to the essential variable, G, the only patterns of behavior that will persist will be those where values

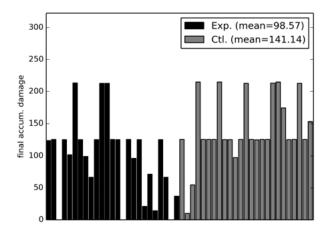


Figure 4: Final accumulated damage for each of 25 experimental and 25 control trials. A higher bar indicates greater damage incurred, i.e. worse performance.

of G are regularly revisited in a way that correlates with the other sensorimotor state variables. In several of the experimental trials, the stable pattern that emerged was one where G was within the viability limits. Why? One possible explanation is that the dynamics of G are more reliable within the viable region than near or outside of it. When G has been inside the viable region for the system's recent history, U=0 and I=0, and thus G changes at a constant rate determined by E. In contrast, when G has recently left the viability limits, one of the hormones will increase in concentration and the way that G changes becomes less correlated with the other sensorimotor variables, (especially given the delay in the differential equations). Thus it seems more likely for the system to find a repeating sensorimotor pattern when it is within the viable region.

For Ashby's homeostat and the robot controllers that it inspired, the stability of the controller has been related to the notion of an $ecological\ invariant$; a relationship with the environment that is maintained by the behavior and that the behavior depends upon. By including the state of G as an interoceptive sensory input, the dynamics of G become part of the habit's sensorimotor "environment" (Buhrmann et al., 2013), such that a behavior must cause G to change in a repetitive way if it is to persist. G is the biological essential variable, but once it is added as a sensory-variable to the IDSM, its dynamics are also essential to the persistence of the habit. The mental and biological autonomous structures are thus intertwined in the sense that they share an essential variable.

Perhaps ecological *invariant* is a bit of a misnomer in this model, as it is not the fixed state of G or any other variable that determines the stability of a pattern of behavior, but more the reliability of a repeated dynamic. In both the control and experimental agents, the behaviors displayed to-

wards the end of the trials are cyclical. This is due largely to the dynamics of the IDSM, where again, only patterns of SM-activity that repeat are reinforced, and as discussed above, only reliable interaction with the environment can result in repeated patterns of SM-activity. At times, the internal dynamics and the environmental interaction are "discordant", in the sense that the motor activity driven by the IDSM does not result in reliable sensory input and the internal dynamics do not "resonate" with the environment in a self-stabilizing manner. As an example of this, consider the more chaotic behavior in Trial E of Figure 3, where the agent is moving around the whole arena, irregularly encountering the feeding area. These irregular sensorimotor trajectories are inherently less stable than those that cause a repeated pattern of sensorimotor state, such as those demonstrated by the subsequent, radially-symmetrical patterns in Trial E, and most of the other agents depicted in Figure 3 (perhaps most apparent in trials A, H, I and J).

In previous attempts to couple biological essential variables to self-maintaining behavior-generating mechanisms, it has been argued that it is necessary to have two nested closed-loops; the first loop being a behavioral coupling between the environment and the organism and the second being an evaluation of the first via an essential variable. such that when the biological essential variable goes out of bounds, the behavior-generating mechanism is reorganized (Di Paolo, 2003). The IDSM-based habits in the experimental scenario are indeed dependent upon the maintenance of a biological essential variable, but we would argue that the two-nested-feedback loop description is not the best way to describe the homeostatic adaptation demonstrated in our model, in that the relationship between the operating limits of the habits and the operating limits of the simulated metabolism are more integrated here than in previous models. The habit does not depend upon the behavior because of a prescribed threshold and response; i. e. it is not due to a random reorganization of the system that is brought about by a pre-specified essential variable going outside of some pre-specified viability limits. Instead, the stability of the behavior and its behavior-generating mechanism is directly dependent upon the repetition of a particular trajectory of the sensorimotor variables, including G. We propose that in this paper we see an example of homeostatic adaptation that blends these two feedback loops into one, suggesting that having two nested feedback loops may not always be necessary.

The system does not always find stability within the viable region, and indeed in several cases (about half) the experimental agents fare no better than the control agents. In these cases, habits have still emerged, but they are unhealthy habits in the sense that they do not maintain the biological essential variable within limits. In ongoing work, and as part of the ALIZ-E project, we are working with our colleagues to develop "diabetic robots" with a simulated

glucose/insulin/glucagon metabolism, that diabetic children can interact with in different ways to investigate how interaction, or modification of the environment could modulate unhealthy habits into healthy habits. By helping the robot transform unhealthy habits into healthy habits, the diabetic children will develop greater self-efficacy, self-confidence and self-esteem, enabling them to better manage their disease (Lewis and Cañamero, 2014). We believe that models such as that presented here can provide insight and fresh perspectives into the relationship between habits and health and how such habits can be better managed.

Conclusion

We have presented our most recent exploration with the IDSM, demonstrating how it can regulate behavior to stabilize essential variables within limits simply by having the state of the essential variable included among its sensors. When a habit emerges in this configuration, it depends upon the dynamics of the biological essential variable, and in this way have demonstrated the possibility of more tightly integrated biological and mental autonomous structures.

Acknowledgments

This research was funded by the European Commission as part of the ALIZ-E project (FP7-ICT-248116). The opinions expressed are solely the authors. The authors would like to thank Tom Froese, Matt Lewis and members of the Embodied Emotion, Cognition and (Inter-)Action Lab for discussions of the research presented above.

References

- Ashby, W. R. (1952). *Design for a Brain: the origin of adaptative behaviour*. J. Wiley, London, 2nd edition.
- Avila-Garcia, O. and Cañamero, L. (2004). Using hormonal feed-back to modulate action selection in a competitive scenario. In *Proc. 8th Intl. Conf. on Simulation of Adaptive Behavior (SAB 2004)*, pages 243–252, Cambridge, MA. MIT Press.
- Barandiaran, X. (2007). Mental life: conceptual models and sythetic methodologies for a post-cognitivist psychology. In Wallace, B., Ross, A., Anderson, T., and Davies, J., editors, *The World, the Mind and the Body: Psychology after cognitivism*, pages 49—90. Imprint Academic.
- Barandiaran, X. E. (2008). *Mental Life. A naturalized approach to the autonomy of cognitive agents*. PhD thesis, University of the Basque Country, Spain.
- Barandiaran, X. E. and Egbert, M. D. (2013). Norm-establishing and norm-following in autonomous agency. *Artif. Life.*, pages 1–24.
- Beer, R. D. (1995). On the dynamics of small continuous-time recurrent neural networks. *Adapt. Behav.*, 3(4):469–509.
- Beer, R. D. (1997). The dynamics of adaptive behavior: A research program. *Robotics and Autonomous Systems*, 20:257—289.
- Brooks, R. A. (1990). Elephants don't play chess. *Robotics and Autonomous Systems*, 6(1-2):3–15.
- Buhrmann, T., Paolo, E. A. D., and Barandiaran, X. (2013). A dynamical systems account of sensorimotor contingencies. *Front. Psychol.*, 4:285.
- Di Paolo, E. A. (2000). Homeostatic adaptation to inversion of the visual field and other sensorimotor disruptions. In *From animals*

- to animats 6: Proceedings of the 6th International Conference on the Simulation of Adaptive Behavior, page 440–449.
- Di Paolo, E. A. (2003). Organismically-inspired robotics: homeostatic adaptation and teleology beyond the closed sensorimotor loop. In Murase, K. and Asakura, T., editors, *Dynamical systems approach to embodiment and sociality*, pages 19—42. Advanced Knowledge International, Adelaide, Australia.
- Egbert, M. D. (2013). Bacterial chemotaxis: Introverted or extroverted? A comparison of the advantages and disadvantages of basic forms of metabolism-based and metabolism-independent behavior using a computational model. *PLoS ONE*, 8(5):e63617.
- Egbert, M. D. and Barandiaran, X. E. (2014). Modelling habits as self-sustaining patterns of sensorimotor behavior. *Submitted to Frontiers in Human Neuroscience*.
- Egbert, M. D., Barandiaran, X. E., and Di Paolo, E. A. (2010a). Behavioral metabolution: Metabolism based behavior enables new forms of adaptation and evolution. In *Artificial Life XII Proceedings of the Twelfth International Conference on the Simulation and Synthesis of Living Systems*, pages 213—220. MIT Press.
- Egbert, M. D., Barandiaran, X. E., and Di Paolo, E. A. (2010b). A minimal model of metabolism-based chemotaxis. *PLoS Comput Biol*, 6(12):e1001004.
- Egbert, M. D., Di Paolo, E. A., and Barandiaran, X. E. (2009). Chemo-ethology of an adaptive protocell: Sensorless sensitivity to implicit viability conditions. In *Advances in Artificial Life, Proceedings of the 10th European Conference on Artificial Life, ECAL.*, pages 242—250. Springer.
- Froese, T. and Ziemke, T. (2009). Enactive artificial intelligence: Investigating the systemic organization of life and mind. *Artificial Intelligence*, 173(3–4):466–500.
- Iizuka, H., Ando, H., and Maeda, T. (2013). Extended homeostatic adaptation model with metabolic causation in plasticity mechanism—toward constructing a dynamic neural network model for mental imagery. *Adapt. Behav.*, page 1059712313488426.
- Kauffman, S. (2000). *Investigations*. Oxford University Press, USA.
- Lewis, M. and Cañamero, L. (2014). An affective autonomous robot toddler to support the development of self-efficacy in diabetic children. In *Proc. 23rd IEEE Intl. Symp. on Human-Robot Interactive Communication (IEEE RO-MAN 2014)*.
- Maturana, H. R. and Varela, F. J. (1980). Autopoiesis and cognition: the realization of the living. Springer.
- Nicolis, G. and Prigogine, I. (1977). Self-organization in nonequilibrium systems: from dissipative structures to order through fluctuations. Wiley, New York, NY.
- O'Regan, J. K. and Noë, A. (2001). What it is like to see: A sensorimotor theory of perceptual experience. *Synthese*, 129(1):79–103.
- Pfeifer, R. and Bongard, J. C. (2006). How the Body Shapes the Way We Think: A New View of Intelligence. The MIT Press.
- Schrödinger, E. (1944). What is life?: The physical aspect of the living cell. The University Press.
- Steels, L. (1995). Intelligence dynamics and representations. In Steels, L., editor, *The Biology and Technology of Intelligent Autonomous Agents*, number 144 in NATO ASI Series, pages 72–89. Springer, Berlin.
- Stewart, J., Gapenne, O., and Di Paolo, E. A. (2010). *Enaction: Toward a New Paradigm for Cognitive Science*. The MIT Press.
- Weber, A. and Varela, F. J. (2002). Life after Kant: Natural purposes and the autopoietic foundations of biological individuality. *Phenomenology and the Cognitive Sciences*, 1(2):97–125.

Active Shape Discrimination with Physical Reservoir Computers

Chris Johnson¹, Andrew Philippides¹ and Philip Husbands¹

¹Centre for Computational Neuroscience and Robotics University of Sussex cj82@sussex.ac.uk

Abstract

We present the first example of 'minimally cognitive' sensorimotor behaviour arising from a body as physical reservoir. By revisiting an experiment introduced by Beer (1996) and replacing the continuous-time recurrent neural network (CTRNN) therein with networks of mass-spring-dampers we demonstrate that bodies may be exploited for more than control and pattern generation and take over some tasks which were previously thought to require a central nervous system.

Introduction

The importance of embodiment in both generating and understanding adaptive behaviour has been increasingly recognised over recent years (Pfeifer and Bongard, 2007). This has resulted in a renewed focus on the form and function of the body. The exploitation of inherent, often passive, dynamics has demonstrated that there is much to be gained, in terms of efficiency and simplification of control, when body-brain-environment interactions are balanced and harmonious (McGeer, 1990; Iida and Pfeifer, 2004; Shim and Husbands, 2012; Zhao et al., 2013). Pfeifer and Iida (2005) have coined the term morphological computation to refer to the way in which a judiciously selected body morphology can be shown to simplify the task of a controller and might be considered to be 'doing' the computational work it had rendered unnecessary. An interesting, and as yet underexplored, extension of this line of thought is to consider how much explicit and active information processing the body might be capable of, further blurring the line between the nervous system and the body. In fact it has already been shown (Valero-Cuevas et al., 2007) that the tendon network of a human finger performs joint torque mode selection in response to varying ratios of tendon tensions: a biological example of explicit morphological computation in action.

This paper describes research intended as a first step towards exploring the information processing potential of networks of simplified muscle-like units acting within an embodied agent engaged in adaptive behaviour. In this work we follow Hauser et al. (2011, 2012), who have reframed morphological computation in compliant bodies as a branch of

reservoir computing (Maass et al., 2004; Lukoševičius and Jaeger, 2009). Hauser et al. (2011) presented networks of mass-spring-damper elements, and showed that with the addition of a simple linear readout these spring networks can perform complex computation requiring non-linear transformation and integration, such as the approximation of filters and inverse kinematics for robot control. These particular networks are of especial interest because they are physically realisable and because of their similarity to various biomechanical muscle models (Hill, 1938; Seyfarth et al., 2002; Baratta and Solomonow, 1990).

In Hauser et al. (2012) it was further shown that when the model was extended to include a feedback loop the networks could also be trained to perform pattern generation without the need for external stimulation. Nakajima et al. (2013) extended the spring network to a biologically-inspired 3D structure and it was shown that this body could also approximate filters and generate limit cycles. Finally, Zhao et al. (2013) replaced the spring network with the body of a spine-driven quadruped robot, referred to as 'Kitty', and used it to generate both locomotion and its own control signals. This robot stands out because the reservoir consists of force sensors embedded within the spine, the element of the body which is actuated, thereby negating any meaningful distinction between body and control.

In the above examples, morphological computation has been demonstrated to make difficult problems such as locomotion both easier and cheaper. However, filtering, pattern generation, and gait control have a character which is more automatic than cognitive. For example, although different gaits may be programmed into Kitty it is still essentially an automaton - its gait may be robust to some variation in the environment but it is incapable of responding to any stimuli which do not reach its force sensors.

We present the first example of 'minimally cognitive' sensorimotor behaviour arising from a body as physical reservoir. We revisited an experiment introduced by Beer (1996) where an agent controlled by a continuous-time recurrent neural network (CTRNN) was shown to be capable of discriminating between objects of different shapes

through active perception. We replaced the CTRNN with a pair of networks based on those introduced by Hauser et al. (2011) and used an evolutionary algorithm to search for valid controllers.

This section has given the theoretical background and introduced the experiment reported upon here. The next section will describe the experiment and the spring network implementation. We will then describe the results obtained before closing with discussion of the results and some future directions for the work.

Methods

Methods are described in three subsections. First the experiment is described, in terms of the agent-environment interaction. Secondly the spring network, the agent's computational core, is described. Finally details of the evolutionary search for valid controllers are given.

The experiment

The simulated experiment is closely based on that described by Beer (1996, 2003). The required behaviour is to dynamically discriminate between a circular object and a diamond-shaped object. Discrimination is manifested as catch and avoidance behaviours for circles and diamonds, respectively. The arena is an area of 400 x 275. Circular objects are of diameter 30 and diamonds have side length 30.

Objects fall straight down from the top of the arena towards the agent with speed 3. In theory both behaviours are tested for at 24 equispaced points in the x-axis interval [-50,50]. However, the use of a symmetrical controller means that only the left-hand 12 tests need be conducted as behaviour on the right-hand side is identical to that on the left. The agent has an antagonistic motor pair aligned to the horizontal axis. The network outputs set the two motor speeds, and the agent's velocity along its axis is the sum of the two opposing motor outputs. The transfer function for the motor pair is given by:

$$5(\sigma(s_r + \theta_r) - \sigma(s_l + \theta_l)) \tag{1}$$

$$\sigma(x) = 1/(1 + e^{-x}) \tag{2}$$

Due to the use of the logistic function σ , each motor saturates at 0 for its minimum and 1 for its maximum. This and the use of a multiplier of 5 for the result of the sum specifies a horizontal velocity in the range of [-5,5]. θ is a constant value which biases the motor activation point.

The agent's sensors are 7 rays uniformly spaced across an angle of $\pi/6$ and centred about the vertical axis. The sensor transfer function is an inverse linear one between the distances of 220 and 0, with its output in the range [0, 10]. Objects are not detected beyond distances of 220. To reduce evaluation time the sensor model was used to construct lookup tables which were then used in the simulation. The

sensor neuron activations lag behind the values of the linear function, as determined by the sensory layer function:

$$\tau_i \dot{s_i} = -s_i + I_i(x, y) \qquad i = 1, \dots, 7,$$
 (3)

where s is the sensor neuron activation, τ is the time constant for the sensor response, I is the sensor function, and (x,y) is the distance vector from sensor to object.

Network states, sensor activations and the position of the agent are all integrated using the forward Euler integration. As in Beer's original experiment an interval of 0.1 is used to integrate sensor activations and the agent's position. In their experiments Hauser et al. used an interval of 0.001 and made use of a solver function to integrate the spring network activity. However the computational cost of such an approach is problematic when evaluating large numbers of candidate controllers in evolution, so a compromise was made here. We found that an interval at least as small as 0.01 is required to achieve stability in the spring model with the parameters used here, so the spring network is integrated 10 times for each 0.1 interval.

Spring networks

Although the elements in these networks are in fact modelled mass-spring damper systems, for the sake of convenience they will henceforth be referred to simply as springs.

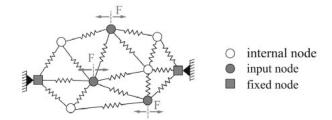


Figure 1: A spring network. The nodes at the opposite ends in the x-axis are fixed while the others are free to move. Some or all of those nodes receive an input as a force in the x-axis. Adapted from Hauser et al. (2011)

The spring networks used here are based upon those in Hauser et al. (2011), illustrated in Fig. (1). The springs are connected to each other in a 2-dimensional plane. Effects such as gravity and friction are neglected in order to simplify the model. The two outermost nodes in a selected axis are fixed while the rest move freely. A subset of the free nodes receive inputs in the form of applied forces. Input forces are applied in a single axis, although this is also for simplification and is by no means a requirement of the model. Reservoir elements were modelled as non-linear springs, defined by the state equations:

$$\dot{x_1} = x_2 \tag{4}$$

$$p(x_1) = k_3 x_1^3 + k_1 x_1 \tag{5}$$

$$q(x_2) = d_3 x_2^3 + d_1 x_2 (6)$$

$$\dot{x_2} = -p(x_1) - q(x_2) + u, (7)$$

where x_1 is the spring extension, k_1 and k_3 represent linear and non-linear stiffness coefficients, respectively, d_1 and d_3 represent the corresponding damping coefficients, and u represents an input force unused in this experiment. In this work we followed the network model of Hauser et al. (2011) in all respects except that the above nonlinear spring model was not used in all networks. In some networks a linear second order spring model was used, with the state equations:

$$\dot{x_1} = x_2 \tag{8}$$

$$\dot{x_1} = x_2$$

$$\dot{x_2} = -\frac{k}{m}x_1 - \frac{d}{m}x_2 + \frac{1}{m}u,$$
(8)
(9)

where k is a stiffness coefficient, d is a damping coefficient, m is the mass on the end of the spring, and, as in Eq. (7), uis an unused input term. For convenience all nodes are given m=1kg. This means that, from Newton's second law of motion, F = ma, forces and accelerations may be treated as equivalent in this network model and Eq. (9) is simplified to a form similar to Eq. (7).

At the beginning of each simulation step the spring extensions are obtained by calculating the distances between the nodes they connect. The rates of change of spring extensions are estimated by the difference between the current extensions and those at the previous step. From these states the instantaneous forces applied to the nodes by the springs can be found, by the use of either Eq. (7) or Eq. (9). The spring forces and input forces are then summed for each node, and the node positions are updated by integration of the resultant accelerations.

Inputs are applied to nodes as horizontal forces, as shown in Fig. (1). In this experiment each network had a total of nine nodes, with two fixed nodes and seven free nodes which each received an input from one of the sensor neurons (see Fig. (2)). An untreated input range of [0, 10] from the sensor neurons was found to give poor results, and so the sensor neuron outputs were scaled and shifted to be in the range [-0.5, 0.5].

The spring network output is a weighted sum of the extensions and extension rates of change of all springs in the network. There is a small departure from Hauser et al. (2011) here. We use the spring extension in the output sum where they used the overall length. The outputs of the two networks are fed into the motor function Eq. (2) in the same way as the CTRNN motor neuron outputs were in Beer (1996).

The CTRNN controller in Beer (1996) was bilaterally symmetric. In this case symmetry of control is achieved by having two identical networks of springs, one of which receives its inputs from the sensory neurons in the reverse

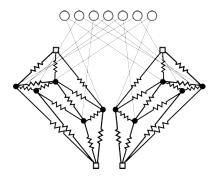


Figure 2: The symmetrical network pair. Two identical networks are connected to the sensor array in opposite orders. The outputs of the two networks are linear sums of the spring states, converted into agent velocity by the use of Eqn. (2). Some nodes and springs are omitted from this diagram for

order to the other. The CTRNN controller consisted of a layer of five fully interconnected recurrent interneurons, and two feedforward motor neurons. The spring network pair replaces these seven neurons.

Searching for solutions

Some network parameters are generated randomly and others are set by a search with a Macroevolutionary Algorithm (MA) (Marin and Sole, 1999). The MA was selected over a Genetic Algorithm (GA) because it was found to be less prone to premature convergence to local optima in the search space for this task. The algorithm was implemented as described in (Marin and Sole, 1999), except for the setting of the two dynamic constants, ρ and τ . The genetic radius for reproduction, ρ , was set by the function $\rho = 0.3(1 - f_{max})$, subject to a minimum value of $\rho = 0.1$. The temperature parameter, τ was set by the function $\tau = (1 - f_{max})$, subject to a minimum value of $\tau = 0.2$. In addition, a constraint was set such that at least one of each generational offspring was randomly generated, in order to promote diversity in later stages. For the same reason, a mutation operator was added such that on average one gene per genotype would be mutated. Mutated genes are moved by an amount in the range of $\pm 10\%$ of the total genetic interval with a probability of 0.9, and replaced with a random value with a probability of 0.1.

A single network topology generated at random at the beginning of each run of the MA is employed by all members of the population. The node coordinates are generated randomly in an area 10 x 10, and then connected with springs by the use of a Delaunay triangulation (Lee and Schachter, 1980). The use of this triangulation method tends to maximise the triangle angles, but also leads to a variable number of springs in the network. Parameters which may be determined by the search are: spring coefficients for stiffness and damping, weights on the sensory inputs to the networks, weights on the spring states for the linear readout, feedback gains, and the bias term for the motor function in Eqn. (2).

For the purpose of evaluation the horizontal distance, d_i , between the agent and the object is clipped to a maximum of 45 and then normalised between 0 and 1. For a catch trial the controller scores $1-d_i$ and for an avoid trial the score is equal to d_i . The final score for a controller is the mean of its individual trial scores. The horizontal distance is clipped to prevent success in one behaviour from dominating a controller's score at the expense of the other.

The MATLAB IDE (The Mathworks, Inc., Natick, MA) was used for all aspects of agent simulation, evolution, and later analysis.

Results

Results are presented in the following order: first, details are given of the performance of the evolutionary search for valid controllers. Secondly, we show some of the inner workings of the network and briefly examine its capacity for memory. We end this section with analysis of two networks by viewing the impact on performance of various lesions.

Searching for solutions

A set of controller features may be enabled or disabled at the beginning of each evolutionary run. These features are: whether to use the linear or non-linear spring model, whether to use spring velocity in the linear readout, whether to evolve real-valued weights on the inputs, whether to use a single random set of spring parameters across the population or to evolve those parameters, whether to employ node position feedback and whether to evolve or to use a constant value for the motor bias in Eqn. (2). The ranges of all evolved parameters are given in Table. (1).

Parameter	Upper limit	Lower limit
Position	10000	-10000
Velocity	10000	-10000
Input weights	-2	2
Linear stiffness coefficients	1	100
Linear damping coefficients	1	100
Non-linear stiffness coefficients	100	200
Non-linear damping coefficients	100	200
Motor function bias	-5	5
Feedback gains	-1	1

Table 1: Limits placed on evolved values.

When node position feedback is employed it is applied as an xy force vector based on a node's displacement from its resting position. Where the motor bias is not evolved a constant value of 2.5640, taken from a successful CTRNN controller which was found when developing the simulation, was used. Where input weights are not evolved, one set of weights is randomly drawn from the set $\{-1,1\}$, and applied to the entire population. These unity weights are of

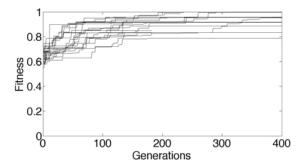


Figure 3: Performance of the MA for a configuration matching controller A, over evolutionary runs of up to 700 generations with a population size of 400. Runs were cancelled when they were either successful or effectively halted. Only the first 400 generations are shown here.

both signs in order to avoid the entire network being pushed in a single direction. When spring parameters were not evolved they were randomised as reported in Hauser et al. (2011). Non-linear spring coefficients are drawn from a uniform distribution in the interval [100,200]. Linear spring coefficients are drawn from a log-uniform distribution in the interval [1,100]. The use of the log-uniform distribution biases samples towards the lower end of the interval.

It was initially unclear which configuration of features was most appropriate, so a set of 20 evolutionary runs, each with a single randomly generated configuration applied across the population, was executed. A population of size 400 was evolved over a short run of 100 generations. It should be pointed out that the MA favours larger populations, but also that the number of the population replaced, and therefore requiring evaluation, upon each generation is variable and typically much less than the population size.

Controller	A	В	С	D	Е
Fitness(%)	99.6	98.9	99.5	98.4	97.8
Number of springs	20	18	18	18	18
Velocity	1	1	1	0	1
Weighted inputs	1	0	1	1	1
Evolve springs	1	0	0	0	1
Nonlinear springs	1	0	0	1	1
Bias motors	1	0	0	0	0
Feedback	0	1	0	1	1

Table 2: Winning combinations. The features of velocity in the readout sum, weighted inputs, evolved springs, nonlinear springs, evolved motor function bias and node position feedback are all optional. 20 evolutionary runs of 100 generations with a population size of 400 were run with random selection of optional features. 5 runs generated successful controllers; each with a unique configuration.

A success threshold of \sim 98% of the perfect score was

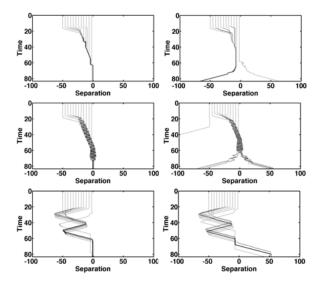


Figure 4: Agent trajectories throughout trials. Trajectories from catch trials are shown on the left and those from diamond trials on the right. Trajectories from three controllers are shown. From top to bottom, controllers A, C and A1.

determined to be sufficient to ensure the correct behaviour, and 5 out of 20 runs resulted in viable controllers. Table. (2) shows the configurations of these 5 controllers. Although this is a small set of results, the variety is striking - the configurations of controllers A and E are similar, but otherwise there is no evidence of a particular configuration which is required for success. However it can be seen that the use of velocity in the linear readout is favoured; being used in 4 out of 5 controllers. Of the remaining features only whether or not to evolve the motor bias stands out; selected in only one result.

Following these results a further 20 evolutionary runs were executed with the arbitrarily selected configuration of controller A. In this case runs continued until the search could be seen to either have succeeded or effectively halted at a local optimum. In this case 4 runs succeeded although others came close. Fig. (3) shows the progress of these runs across the first 400 generations. As yet it is unclear as to whether the difficulty of the problem or the character of the MA is more responsible for the number of failures. However, since evolutionary search is merely being used as a method to find a viable solution, and it readily finds several in our batch runs, its efficiency is not a major concern at present. One of this second set of results, referred to as A1, appears in figures and analyses throughout this section.

Successful networks have proven to be diverse in their topologies as well as their configurations. As shown in Fig. (5) success does not seem to require any particular form of network.

Finally, from a high-level point of view, it can also be seen

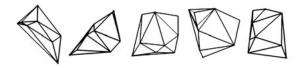


Figure 5: Network topologies of successful controllers. Controllers A through to E are shown from left to right.

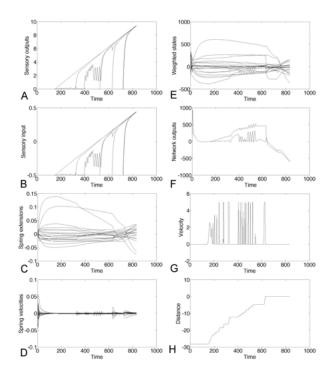


Figure 6: Plots of inputs, states and outputs of controller A from a catch trial. All plots are over time. Plot A shows the outputs of the sensory neurons. Plot B shows the shifted and scaled sensory input to the networks. Plot C shows the spring extensions of one network. Plot D shows the spring velocities of the same network. Plot E shows the weighted positions added to the weighted velocities. Plot F shows the outputs of both networks. Plot G shows the velocity of the agent. Plot H shows the distance between the agent and object in the x-axis.

that various different strategies are possible. The trajectories of controller A, C and A1 for all 24 trials are shown in Fig. (4). Controller A inches towards objects until it can distinguish between them, controller C finds objects quickly and then oscillates around their position until making a decision, and controller A1 scans back and forth.

Network analyses

Fig. (6) illustrates some of the workings of controller A in making a successful catch. As mentioned previously, the sensory inputs are scaled and shifted to the range [-0.5, 0.5].

This has the effect of biasing the network to activity, with the springs already in motion for the first 10s even though the sensory neurons have zero output. The spring extensions and velocities are combined in the readout in this controller. Due to the very large weights used in the readout the network outputs are similarly high. Given that the motor function includes the saturating logistic function, this leads to a motor behaviour of rapidly switching the motors on and off, resulting in the agent creeping incrementally towards the position of the falling circle.

An interesting observation is that as long as the sensory neurons all have zero output, so do the networks. Given that the spring positions are non-zero in this initial period, it is clear that the linear readout is balanced to not respond to this quiescent activity. This is perhaps appropriate to reactive behaviour, but the possibility remains for evolution to lead to a more proactive search strategy by generating an imbalanced readout.

We next turned to measuring the memory of the network following a method based on that described by Maass et al. (2004). Maass et al devised an input stream with zero mutual information (and therefore also zero correlation) between different segments. Then, to obtain a measure of network memory, readout neurons were trained to reproduce segments of the input stream from earlier periods, and segments of the output signal were correlated against the input segments they should have reproduced. A similar approach is taken here, although we chose not to measure general memory capacity but rather to see if these networks can retain information of the inputs they are evolved to deal with. For this reason the readout was trained to recover the simplest combination of the input signals over the course of a single trial, their sum into a single time series. In this case many segments of the input stream have high correlation with one another as, due to the tendency of agents to position themselves under an object until it can be recognised, the general trend is for sensory input to increase as a ramp. Therefore, as a baseline, the same series of correlations was performed for input segments against one another. Where the readout shows no more correlation with earlier input stream segments than its corresponding input segment does, then there can be considered to be no memory. On the other hand, a stronger correlation between the output of the readout and the delayed input it is trained to reproduce than between input and delayed input is indicative of memory in the network. The same test is performed for readouts which receive only spring extensions as inputs, readouts which receive only spring velocities, and readouts which receive both.

The results of some of these tests are shown in Fig. (7). It can be seen that, in general, spring extensions encode more relevant information than spring velocities, but that the combination of the two encodes more than either alone. Controller A shows no convincing sign of memory. For the avoid

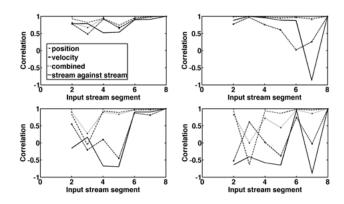


Figure 7: Measuring memory. The linear readout is trained to recover inputs from 10s ago. Then the input stream and the controller output are split into 10s segments and the correlation of each output segment with the prior input segment is calculated. Plots on the left are from a single catch trial and those on the right are from an avoid trial. The results for two controllers are shown. The top row is for controller A, and the bottom row is for controller A1.

trial there is a period where the correlation of input segment against input segment is of the opposite sign to that of readout segment against input segment, but the magnitudes are roughly equal. This seems consistent with the behaviour of this controller. As shown in Fig. (4) for both catch and avoid behaviours, initially this controller gradually creeps towards the object as it falls, suggestive of a purely reactive network. The result for controller A1, however, does indicate a degree of memory effect, with the network being able to recover more information about earlier input segments than the input stream itself. Once again, this is consistent with the general strategy - unlike other results this controller drives away from the object and then returns to it, a behaviour which seems of a more proactive character and implies memory of at least which side of the agent the object is on. It should be pointed out that this controller does not make use of feedback. Any present memory is only transient, fading mem-

Analysis of a network with such complex dynamics in a sensorimotor loop is far from trivial, but a certain amount can be discovered by recording changes in behaviour as parts of the network are disabled. Four experiments, illustrated in Fig. (8) for two controllers, were conducted. In the first three experiments changes were made to the springs, one at a time, and the performance scores for the modified network for all 24 trials were recorded. In the fourth, one input at a time was disabled and the performance scores were recorded. In order, the modifications for springs were: to disconnect them from the linear readout, to remove them from the network completely, and to disable their non-linearity.

Various observations can be made from these plots. It

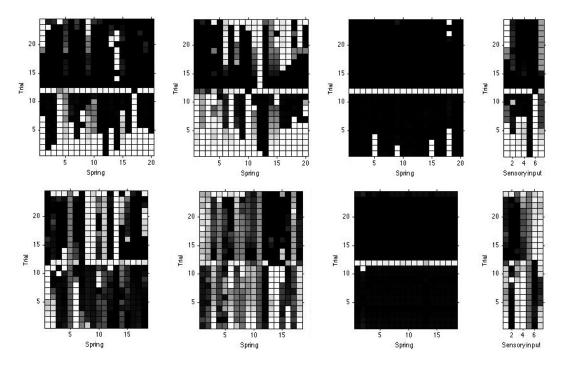


Figure 8: Lesion experiments. The top row of plots shows results for controller A and the bottom for controller A1. The rows in plots show the performance on a trial by trial basis. The brightness of a grid element shows the effect of the lesion. Black regions show unaffected performance and lighter regions show impaired performance. Trials 1 to 12 are catch trials and trials 13 to 24 are avoid trials. From left to right: one spring at a time is disconnected from the readout; one spring at a time is removed from the network, one spring at a time is linearised; one input at a time is disconnected.

can be seen that adjustments to the network for controller A tend to cause failure in catching circles far more often than in avoiding diamonds, as though this controller is predisposed to avoidance. To support this conclusion, this agent also shows low dependence on all but the outermost sensors for avoidance. Complete removal of springs from the network causes a lot of failure in both agents. This is no surprise, given the tightly coupled nature of the network dynamics. Neither controller shows a strong dependence on spring non-linearity,

The plots in Fig. (8) suggest that for both of these agents the most difficult trial is the last where catching behaviour is required. This is surprising as at the beginning of this trial the object is only slightly offset from the agent's position. The reason for this has not yet been properly uncovered, but it seems probable that it is connected to the large weights in the linear sum as relatively small differences in sensory input are amplified into high velocity, which could lead to a sudden loss of the object's position.

Discussion

Discussion of results

The CTRNN described by Beer (1996) was bilaterally symmetric and fully interconnected in the interneuron layer.

Symmetry is achieved here by the use of a pair of identical networks, coupled only indirectly by their roles in the sensorimotor loop. The overall behaviour of the agent is therefore the result of complementary activity in the two sides of the simulated body.

Successful agents obtained by evolution pass the bar for autonomous task-based behaviour, capable of both responding directly to stimuli and making use of short-term memory to guide their motion. The behaviour of these agents goes beyond the maintenance of locomotive gaits to the selection of an appropriate action based on active integration of sensory information. This more complex behaviour is also achieved with a relatively small number of springs. Hauser et al. (2011) used a network of 78 springs; these controllers contain approximately 40.

The morphological computation evident in the tendon network of the human hand is a striking example of parsimony in evolution. This work lends support to the hypothesis that bodies can be and will be doing some of the computational work which used to be considered the sole dominion of the central nervous system.

The use of an evolutionary algorithm to obtain valid controllers has led to strong indications that the domain of these networks is rich in its variety of computational resources.

This potential for diversity may have been less apparent if a learning algorithm was used.

Magg and Philippides (2006) have already shown that this problem may be solved with a non-dynamical ANN, and, as in controller A, it appears that most results seen so far employ little or no memory. However, controller A1 is an interesting result which does seem to rely on memory and suggests that behavioural tasks of a more challenging character than this one may therefore also be addressed with similar spring networks.

Further work

The very large weights used in the linear readout sum tend to lead to the agent switching between extremes of velocity. In a real-world system this would be inefficient and probably lead to shortened motor life. At present we are examining how far the scale of those weights may be reduced while still achieving valid controllers. First results are encouraging, with signs of smoother behaviour.

The results obtained so far indicate that due to the tight coupling throughout the networks they will not fail gracefully when damaged, as indicated in Fig. (8). However, it may be that a damaged network may be easily retrained to recover its function. In future work we will examine the efficacy of retraining by submitting impaired controllers back to evolution.

Although enough good and varied results were found to convince that these paired networks may be used to effect reactive behaviour, efforts are underway to both improve and tune the evolutionary algorithm in use and to make the networks more evolvable. As far as the second point goes, one feature of the problem which has not yet been placed under evolutionary control is the network topology. We believe that topological design by evolution will lead to an improved success rate.

Later projects will explore the capability of spring networks to generate behaviour in compliant robots, and to deal with real-world, noisy, situations.

References

- Baratta, R. and Solomonow, M. (1990). The dynamic response model of nine different skeletal muscles. *Biomedical Engineering, IEEE Transactions on*, 37(3):243–251.
- Beer, R. D. (1996). Toward the evolution of dynamical neural networks for minimally cognitive behavior. From animals to animats, 4:421–429.
- Beer, R. D. (2003). The dynamics of active categorical perception in an evolved model agent. *Adaptive Behavior*, 11(4):209–243.
- Hauser, H., Ijspeert, A. J., Füchslin, R. M., Pfeifer, R., and Maass, W. (2011). Towards a theoretical foundation for morphological computation with compliant bodies. *Biological cybernetics*, 105(5-6):355–370.

- Hauser, H., Ijspeert, A. J., Füchslin, R. M., Pfeifer, R., and Maass, W. (2012). The role of feedback in morphological computation with compliant bodies. *Biological cybernetics*, 106(10):595–613.
- Hill, A. (1938). The heat of shortening and the dynamic constants of muscle. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 126(843):136–195.
- Iida, F. and Pfeifer, R. (2004). Cheap rapid locomotion of a quadruped robot: Self-stabilization of bounding gait. In *Intelligent Autonomous Systems*, volume 8, pages 642–649.
- Lee, D.-T. and Schachter, B. J. (1980). Two algorithms for constructing a delaunay triangulation. *International Journal of Computer & Information Sciences*, 9(3):219–242.
- Lukoševičius, M. and Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149.
- Maass, W., Natschläger, T., and Markram, H. (2004). Computational models for generic cortical microcircuits. *Computational neuroscience: A comprehensive approach*, pages 575–605.
- Magg, S. and Philippides, A. (2006). Gasnets and ctrnns–a comparison in terms of evolvability. In *From Animals to Animats 9*, pages 461–472. Springer.
- Marin, J. and Sole, R. V. (1999). Macroevolutionary algorithms: a new optimization method on fitness landscapes. *Evolutionary Computation*, *IEEE Transactions on*, 3(4):272–286.
- McGeer, T. (1990). Passive dynamic walking. the international journal of robotics research, 9(2):62–82.
- Nakajima, K., Hauser, H., Kang, R., Guglielmino, E., Caldwell, D. G., and Pfeifer, R. (2013). A soft body as a reservoir: case studies in a dynamic model of octopus-inspired soft robotic arm. Frontiers in computational neuroscience, 7.
- Pfeifer, R. and Bongard, J. (2007). How the body shapes the way we think: a new view of intelligence. MIT press.
- Pfeifer, R. and Iida, F. (2005). Morphological computation: Connecting body, brain and environment. *Japanese Scientific Monthly*, 58(2):48–54.
- Seyfarth, A., Geyer, H., Günther, M., and Blickhan, R. (2002). A movement criterion for running. *Journal of biomechanics*, 35(5):649–655.
- Shim, Y. and Husbands, P. (2012). Chaotic exploration and learning of locomotion behaviors. *Neural computation*, 24(8):2185–2222.
- Valero-Cuevas, F. J., Yi, J.-W., Brown, D., McNamara, R. V., Paul, C., and Lipson, H. (2007). The tendon network of the fingers performs anatomical computation at a macroscopic scale. *Biomedical Engineering, IEEE Transactions on*, 54(6):1161– 1166
- Zhao, Q., Nakajima, K., Sumioka, H., Hauser, H., and Pfeifer, R. (2013). Spine dynamics as a computational resource in spine-driven quadruped locomotion. In *Intelligent Robots and Systems (IROS)*, 2013 IEEE/RSJ International Conference on, pages 1445–1451. IEEE.

Hormonal modulation of development and behaviour permits a robot to adapt to novel interactions

John Lones¹, Matt Lewis¹ and Lola Cañamero¹

Embodied Emotion, Cognition and (Inter-)Action Lab School of Computer Science & STRI, University of Hertfordshire College Lane, Hatfield, Herts, AL10 9AB, U.K. J.Lones@herts.ac.uk, matt-l@semiprime.com, L.Canamero@herts.ac.uk,

Abstract

Hormones are known to play a critical role in modulating the behaviour and development of organisms when confronted with different environment challenges. In this paper we present a biologically plausible hormonal mechanism that allows an autonomous robot to interact appropriately with novel objects and interactions depending upon both its current internal state and its past experiences. In our experiments, robots that had been exposed to negative experiences during their initial developmental phase displayed withdrawn behaviour and were less likely to explore new objects and environments, or to engage with a human caregiver. In contrast, robots with a positive upbringing showed much greater levels of outgoing behaviour such as exploration and social interaction.

Introduction

Hormones have been established as critical behavioural modulators for many types of organisms ranging from vertebrates (leRoith et al, Montoya et al 2012) right through to simple unicellular organism (Kabara 1988) highlighting them as a potential fundamental aspect of life. Like in biological organisms, hormones have also been demonstrated to have a potentially critical role to play in autonomous robots. In these artificial models, hormones have been utilised to successfully modulate different action selection mechanisms in order to achieve appropriate behaviour in a range of scenario (Avila-García & Cañamero, 2004, French & Cañamero 2005, Blanchard & Cañamero 2006, Chelian et al 2012, Krichmar 2013). In our own previous studies (see Lones & Cañamero 2013 & Lones et al 2013 & Lones et al 2014) we integrated hormonal modulation of a homeostatic system through an epigenetic mechanism, in which hormones were secreted in relation to homeostatic deficits. This mechanism enabled a sensory-driven autonomous robot to adapt successfully to a wide range of different environmental challenges and gave rise to the emergence of unique behavioural characteristics.

In this paper, we look to improve the adaption capabilities of an autonomous robot by endowing it with an hormone signalled epigenetic mechanism that modulates the development of a wide range of survival-related and social behaviours as a function of both its internal state and the environmental conditions, and that would permit the robot to interact appropriately with novel objects in its environment. We do this by adding two new hormones to our previous system, functionally akin to two chemical modulators, the steroid hormones corticosteroids and testosterone.

In biological organisms both hormones have long drawn particular interest for their role in modulating a wide range of value-laden survival and social behaviours. This occurs due to the interaction between the hormones and one of their primary targets, the amygdala (Koolhass et al., 1990). While the exact mechanisms are unknown, once the hormones have reached the amygdala, their behaviour is better understood. Testosterone is linked to promoting outgoing reward-seeking behaviours such as dominance, aggression, exploration, and curiosity (Daitzman & Zuckerman, 1980, Mazur & Booth 1998). In contrast, corticosteroids are related to avoidance and withdrawal behaviours (Buss et al., 2003; Montoya et al., 2012). Moreover, these hormones do not only modulate emotional processing of the amygdala but are also believed to affect its neural connectivity to other areas of the brain, particularly the orbitofrontal cortex. Exposure to corticosteroids leads to strengthening these neural connections, while testosterone weakens them (Mehta, 2010; van Wiggan et al., 2010). As the orbitofrontal cortex is associated with decision making, strengthening or weakening the emotional input from the amygdala could result in additional behavioural modulation (Bechara, 2000)

Although the two steroid hormones have significant potential to modulate behaviour, actual studies into the individual roles of these hormones do not always offer conclusive evidence. This is particularly noticeable in human studies where results are normally limited to observation of subjects, which can even be contradictory. There are at least two likely explanations for this. Firstly, there is significant evidence to suggest that both cortisol (CHT) and testosterone (T) work in tandem to modulate behaviour and it is the ratio or imbalance between both chemicals that is important (Montoya, 2012). For example, in a situation with a high T/CHT ration (high T low CHT) aggression is more prevalent than in a situation

with an equal ratio, even when the T level remain constant (Popma et al., 2007).

Secondly, the effects of hormones and secretion are likely to be subjective. This is particularly relevant for organism with highly complex cognition and neural mechanisms, such as humans. Where aspects such as learning, planning, normative behaviour and beliefs gained through life-long experience will affect "consciousness" and therefore can lead to differences in individual emotional processing (Arnold, 1960, LeDoux, 1993 & Khalin, 1993). However, it is not only the level of neural mechanisms that can lead to subjective hormonal modulation. More recently, evidence has arisen of phenotypical plasticity in the neuroendocrine systems responsible for the secretion and regulation of T and CHT. Changes in gene expression occurring within these neuroendocrine systems are known to be associated with extreme forms of behaviour (Mcgowan et al 2009). However, it is also highly likely that these changes could have an effect on day-to-day behaviour.

The neuroendocrine systems of T and CHT consist of the Hypothalamic-pituitary-gonadal axis (HPG-Axis) and the Hypothalamic-pituitary-adrenal axis (HPA-Axis) for T and CHT secretion respectfully. While these two axes are often considered separate entries, they are interconnected through feedback loops. Specifically, research has shown that the HPA-Axis supress the activity of the HPG-axis on all levels. In addition the HPA-AXIS contains a negative feedback loop that consists of glucocorticoid receptors which, in response to rising corticoid levels, signal for the suppression of the axis activity (Montoya et al 2012).

However, this is not a simple static relationship between cortisol levels and HPA activity. Research has suggested that the glucocorticoid receptors responsible for the feedback are susceptible to *epigenetic changes* consisting of upwards and downwards regulation. Downwards regulation, which is a reduction in the total number of receptors, leads to reduced sensitivity to corticoids and thus weakens the negative feedback loop. In contrast, upwards regulation leads to an increased number of receptors and therefore increased sensitivity and a more reactive negative feedback loop (Liu et al 1997 Mcgowan et al 2009, Zhang et al 2013).

Downregulation of glucocorticoid receptors has been linked with, and believed to be triggered at least partially by, continuous high levels of corticoids in the system (Mcgowan et al 2009). Upregulation, on the other hand, has been associated with positive upbringing and experiences during early life with dopamine considered a potential chemical trigger (Liu et al 1997). Much like many other forms of epigenetic changes organisms tend to be more susceptible to mechanism during critical periods of development (Reik et al 2001).

In this paper, we present a hormone-driven biologically plausible robotic model of these mechanisms. Like in the

biological examples discussed, in this artificial model hormones play a critical role in short-term modulation of both the internal environment and the connectivity of different "neural functions" of the robot. This is achieved by secreting the hormones as a function of the robot's internal state and external stimuli, creating a "chemical soup" that surrounds the robot's neural functions. Each "neural function" has receptors that can detect the concentration of specific hormones and are susceptible to modulation accordingly. In addition, hormone levels are also used to signal epigenetic changes in specific areas of the model during critical developmental periods, causing long-term implications on the behaviour of the robot arising from specific characteristics of its internal state and of the environment stimuli that it was exposed to during the critical developmental period.

Robotic model

To test the viability of the previously described mechanism experiments were run utilizing the koala II robot. We added a webcam to the standard robot configuration, which consists of 14 IR range sensors. Control of entire model was handled through a serial connection to a computer running Ubuntu, the architecture was programmed in C++, and openCV was used for vision and image processing. Vision was used to detect resources based on predetermined characteristics such as shape and colour.

As we will discuss in more detail later, we conducted experiments in a noisy environment in open space in our lab. Due to this addition of noise we found it essential to add two functions between the IR-sensors and the Action selection mechanism. The first function consisted of a form of sensory memory similar to iconic memory in which information from IR's are briefly stored before decaying.

$$\begin{cases} Tsv_{i\,t} = \frac{sv_{i\,t-1}}{r_D} & if \ sv_{i\,t} < Abst_i \\ Tsv_{i\,t} = \ sv_{i\,t} & otherwise \end{cases} \tag{1}$$

where Tsv is total IR sensory value which is passed to the ASM, sv the current sensor value, i is sensor number, t time, rD is the rate of decay which was set to make sensory information fully decay in just under a second, and Abst the absolute threshold that defines the minimum value that sensors need to become 'active'. The second function heightens the sensitivity of sensors neighbouring an active sensors by reducing the Abst by heightened value (hV) which is equal to 33%

Homeostatic variables

The architecture of the robot includes three survival-related variables which are homeostatically controlled: health, energy and temperature. These decrease as a function of the robot's actions and interaction with the environment. Whereas 'health' is simulated, the two other variables are linked to the actual physics of the robot. When the robot detects contact of significant strength, its homeostatic variable 'health' decreases proportionally to the size of the force. This deficit can be recovered by finding and utilising a repair resource. Energy is linked to the robot's battery, which has a maximum capacity of 3.5 ah (3500 mAH), permitting around 4 hours of usage. Due to the long battery duration, to make the environment more challenging, the architecture was programed to allow the robot sense a maximum of 75 mAh of charge at any time, equivalent to a running time of around 5 minutes at a moderate activity level. Consumption of an energy resource increases the charge available to the potential maximum of 75 mAh. Finally, temperature is related to the speed of the motors and directly sensed using the internal temperature monitors of the robot. Each of the survival-related homeostatic variables has a lethal boundary which if transgressed results in the agent's death. In the case of energy and health, the lethal boundary is set at the bottom end of the range of permissible values, in the case of temperature the lethal boundary is at the upper end of the range.

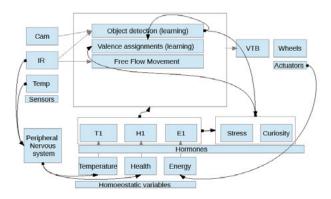
Table 1 the homeostatic resources

Homeostatic Variable	Energy	Health	Temperature			
Derived from	Battery level	Simulated	Internal sensors			
Range	0-75 mAh	0-100%	26–32 degrees			
lethal boundary	0 mAh	0%	32 degrees			
Deficits	Semi- dynamic	Dynamic	Dynamic			
occurrence						
recovery	Energy	Health	Reducing			
	Resource	Resource	motor speed			

Action selection architecture Mechanism (ASM)

The robot's action selection architecture consists of three main internal systems: object and stimuli detection, valence assignment and free flow movement. Behaviours emerge from the activation, interaction and modulation of the different internal systems, rather than being pre-determined.

Figure 1: Action selection architecture used in this paper.



Object detection is achieved by a combination of image and IR data. Data from the camera is processed (using the openCV library) to detect resources scattered around the environment. The IR sensors feed into a learning growing artificial neural network in order to detect and classify novel objects. The latter uses primarily information about the size of objects, determined by the number of IR sensors active. Since this type of neural network needs a very long training phase to learn from scratch, for the experiments presented in this paper we trained the network off-line; the same network was used for all runs

Table 2 The types of objects the IR sensors can detect

Number of sensors active	Type of object
0	Empty space
1	Small object
2	Medium sized object
3	Large object
4+	Wall
0, but both neighbouring neurons active	A hole or gap

As well as being able to classify different objects into size groups, a second function of this system is to *predict* the behaviour of a detected object in terms of how close or far it will be at the next point in time, or which sensors might detect it next if it is being passed. Each IR sensor can store up to three predicted locations at any one time – those generated by itself and by its two neighbouring sensors. Since, as we will see later, the environment is primarily static, the robot will always assume any change in distance to be the result of its own movement. For example, while moving forward, an object that is detected in front of the robot will be expected to be closer, while an objected detected on the left will be expected to be later detected by an IR sensor on the same side but further away in the direction of motion, and therefore the predicted location of this object is shared with this sensor.



Figure 2: Object prediction system. The distance to the predicted location is determined by a range of values between two boundaries, which are calculated as follows:

$$upperBoundary_i = Tsv_i + oS_k + sP_i + H_c - H_s$$
 (3)

$$lowerBoundary_{i} = Tsv_{i} \frac{upperBoundary}{1 + (\frac{4}{sP_{i}})}$$
 (4)

where sP_i is the speed at which the robot is moving towards the object, oS_k the expected object speed (always 0 in this case as objects are expected to be stationary) the current concentration of stress (H_s) and curiosity (H_c) hormones

The width of the predicted area (if the prediction is passed on to a neighbouring sensor) is determined by

$$\begin{cases} passed \ to \ left \ \ if \ \ \frac{lWSpeed}{rWSpeed} \ + oS_k > dS_i \\ passed \ to \ right \ \ if \ \ \frac{rWSpeed}{lWSpeed} \ \ + oS_k > dS_i \end{cases} \tag{5}$$

where IWSpeed and rWSpeed are the speed of the left and right wheels, respectively and dS_i is the distance between neighbouring sensors, which is unique between each pair.

If no object is detected by a sensor or its neighbours, then the robot will only expect to detect distant objects. If an object is detected outside the predicted area, e.g., if an object is placed directly in front of the robot, it will be perceived as unexpected and treated as a stressor.

Valence assignment occurs after an object has been detected and its value depends upon both the type of object and the current internal state of the robot. Valence is an emotion dimension associated with objects, interactions or events (Russell, 1980; Posner et al, 2005) and that provides values along a pleasure-displeasure continuum. The value of objects is calculated as a function of the potential threat associated with the object, and the internal state of the robot. The base valence for a novel object (V_o) is determined as a function of their size – larger objects are perceived more negatively as they could potentially be more dangerous,. The internal factors contributing to valence are the current concentration of stress (H_s) and curiosity(H_c) hormones. The effect that these hormones have on the perception of different objects scales differently (S_{oh}) for each object as shown in table 3.

$$valence = V_o + (H_s S_{os}) + (H_c S_{oc})$$
 (6)

Table 3 Valence detection

Type of object	Initial	Curiosity	stress
	Valence	scale	scale
Empty space	1	1	-1
Small object	200	2	-4
Medium sized object	-200	8	-8
Large object	-400	14	-12
Wall	0	-2	10
A hole or gap	-800	20	-16

Hormone model

The epigenetic artificial hormones in this model consist of two main groups, that we have named "endocrine hormones" (eH) and "neuro-hormones" (nH). While both group types share common characteristics, they also present some significant differences. Firstly, endocrine hormones (eH) are secreted by their respective glands in relation to homeostatic variables deficits. Each variable has an associated hormone – E1 for energy, H1 for heath and T1 for temperature – released by a corresponding gland. The secretion of each of these hormones occurs when the ASM stimulates its associated glands (g_i) as a function of the relevant homeostatic deficit.

$$g_i = x_i \varphi_i^{defict} \tag{7}$$

where x_i is the strength of stimulation from the ASM and φ_i the gland's activity level.

The gland's activity level is determined by a simple biologically plausible epigenetic mechanism akin to epigenetic adaption that we previously implemented in (Lones & Cañamero, 2013) This mechanism consists of a positive feedback loop were high concentration of a secreted hormone will lead to increased gland activity.

$$\varphi_i = \varphi_i + \frac{eH_i}{e^S} \tag{8}$$

where eS is a constant value used to control the speed of the epigentic change.

The secretion of the second group of hormones is triggered by a combination of the robot's external perception of the environment and its current internal state. This hormone group includes two hormones called "stress" and "curiosity". As the name suggests, the stress hormone (H_s) is inspired from the roles that cortisol and serotonin play in regulation of stress responses, and hence it regulates stress response in the robot.

$$g H_s = roD + \left(\frac{oS + F}{x}\right) * fL \tag{9}$$

where oS is overstimulation (a simple addition of all objects detected), F is fear (total value of negative stimuli detected), fL is the feedback loop, and x a constant, roD $(0 \le roD \le$ 1) is the robot self-perceived "risk of death" (Risk of death as a measure of viability was previously developed by Avila-García & Cañamero ,2004) which is derived from the current concentration of the three endocrine hormones. The risk of death is determined by both the current level and duration of any homeostatic deficits. In addition, since hormone concentration is also affected by the previously described epigenetic mechanism, the perceived risk of death will also be dependent on the development of the robot. For example, whereas a robot that is constantly low on energy will likely develop hyper-sensitivity to energy deficits and will end up perceiving a high risk of death when energy deficits occur, a robot that "grows up" in an environment that permits it to always maintain high energy levels will develop a natural tolerance to deficits. The stress hormone is secreted under three types of circumstances: when there is a high risk of death (linked to homeostatic deficits), when there is overstimulation from exposure to different novel objects, and in the presence of perceived threats.

Our curiosity hormone (H_c) takes inspiration from the roles that hormones such as testosterone and dopamine play in regulating (in this case increasing) behaviors related to dominance, aggression and curiosity.

$$g H_c = \frac{(1-roD) + \left(\frac{R+pS}{x}\right)}{1+H_S} \tag{10}$$

As can been seen, H_c secretion occurs with low risk of death, perception of interesting objects (pS), and as homeostatic deficits are recovered (R). In addition, the concentration of H_s limits the secretion of H_c in a similar manner to the biological interaction between cortisol and testosterone through the HPA axis. Much like the HPA axis found in biologically systems, an epigenetic mechanism exists in our artificial HPA negative feedback loop. In biological systems, this mechanism consists of glucocorticoid receptors present at both the hypothalamus and the anterior pituitary lobe of the pituitary gland. These receptors detect the current concentration of corticosteroid and lead to a negative feedback loop if levels are too high (Liu et al 1997 Mcgowan et al 2009, Zhang et al 2013). An important feature of these receptors is that they are susceptible to epigenetic changes in gene expression. Exposure to constant high levels of corticosteroids, for instance, leads to downregulation in the feedback loop (Mcgowan et al 2009). However positive environmental upbringing has shown to lead to upregulation. While the exact mechanic of upregulation is unknown, exposure to hormones/neuro transmitters associated with happiness and outgoing behaviour i.e dopamine are potential and realistic mechanism with some supporting evidence (Liu et al 1997). In this model, upregulation and downregulation of the glucocorticoid receptors and therefore the regulation of the feedback loop (fL) are respectively associated with exposure to H_s and H_c .

$$fL = \frac{1}{1 + e^{-(H_S - H_C)}} \tag{11}$$

Once part of the chemical soup (cS_h) all hormones decay at the same rate which in this model was set to 0.95

$$\sum_{h=0}^{4} cS_h = cS_h * 0.95 \tag{12}$$

Experiments

For this paper we tested this architecture under two main experimental conditions: a relatively static environment where the only changes occurred as a result of the interactions of the robot, and a much more dynamic environment that included human-robotic interaction. The environments used for our tests were implemented in the open space of our lab, an area of about 45 m² in which desks and chairs are located near the walls, surrounding an empty central area. The robot could roam freely around the lab that was only modified by removing the swivel chairs and by placing limited plywood boards to protect sensitive or delicate areas. Resources, obstacles and other environmental stimuli were then added as can be seen in figure 3. In all cases, an identical architecture and robot was used for each experiment.



Figure 3: The positive and negative environments. Figure 3 shows two different angles of the environment the positive on the left and negative environment on the right with the differences discussed in more detail bellow

In both experimental conditions ("static" and "dynamic" environments), the tests involved two phases. In the first phase, identical for both experimental conditions, the robot spent its critical developmental period (in our case the first 5 minutes of its "life") in one of the two environments previously discussed, with an equal split between the two types of environments within each experimental condition: either a "positive" or easy environment (half of the runs for each experimental condition) or a "negative" or stressful environment (half of the runs for each experimental condition). As can be seen in Figure 3, both positive and negative developmental environments share similar design with a few subtle but important differences, as follows. A) the first difference is the homogeneity of materials used, with more variation occurring in the negative environment. As IR-Sensors naturally respond differently to different materials. increasing variation naturally leads to more fluctuations in sensor readings. Due to the previous described neural mechanism, these fluctuations are likely to lead to a stress response. B) The second difference is the increased use of objects made of textures and/or colours particularly difficult to detect in the negative environment. Not only does this lead to fluctuations in sensor readings but also, since the robot is unable to accurately detect distance, it also increases the likelihood of collisions. C) The third difference is the spacing between objects. In the negative environment, distance between different objects is small, increasing the potentially for over-stimulation. In addition, smaller spaces in structures like the maze depicted in the figure also hamper navigation significantly, potentially increasing stress. D) The Final, a fourth difference is the "reward" obtained for exploring the environment, which is greater in the positive environment. For example, in the positive environment the reward for completing the maze is an easily accessible homeostatic resource. In contrast, while the resource is still present in the negative environment, the likelihood of finding and accessing the resources, and therefore of getting a reward for completing the maze, is smaller.

The second phase differed for each condition. A set of 10 single-robot runs, each of duration of 15 minutes, took place in relatively static environment, and a set of 10 runs of duration of 5 minutes per run in the dynamic environment. In this second condition, human-robot interaction took place in an empty environment after the robots had developed. While challenges and potential stimuli to explore were still in abundance at the edges of the environment, the centre was largely barren in order to increase the chances that the robot would focus on human interaction. Runs were reduced to 5 minutes since the experimenter could finely control exposure to stimuli, making unnecessary the longer duration of runs that was used in the static environment condition in order to ensure that the robot could fully explore the environment.

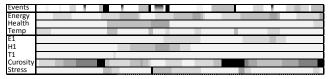
While the ASM was never specifically designed or programed for human-robot interaction, we found that it would naturally lend itself to simple types of interaction. For example, the robot generally found stroking interaction positive as it led to a release of the curiosity hormone, whereas sudden movement usually invoked a fearful or negative response. The robot's response to an action was also highly influenced by the way it was carried out. For example like in (Cañamero & Fredslund, 2000), the speed, force and duration of the stroking motion will have a significant impact on the robot's response The response is again also influenced by a combination of the robot's internal state and developmental history. A robot that has had extensive human interaction was more likely to enjoy vigorous stroking where as one that has suffered extreme deprivation might find displeasure in even minimal contact.

Experimental results and discussion

To analyse the results of our experiments in terms of the performance of the robot, we had initially planned to use the viability-based indicators of performance developed in previous work (Avila-García & Cañamero, 2004.), such as wellbeing, overall comfort, or risk of death. However, due to the epigenetic developmental mechanism in our present architecture, the robots that developed under different environmental conditions developed different tolerance to stimuli and homeostatic deficits. These differences in tolerance led to skewed results using the above-mentioned performance indicators, as will discuss below.

Experiments in the static environment. As can be seen in figure 4, the interaction between the environment and epigenetic hormonal mechanisms made the robot have significantly different behaviours once developed and placed within the "neutral" testing environment.

Internal state of a robot developed in the positive environment



Internal state of a robot developed in the negative environment



Figure 4: The internal state of two robots. Figure 4 shows the internal state of a robot from the positive and negative environments during the 15 minutes in the static environment. Darker colours indicate increased stimuli, higher homeostatic deficits or increased hormones concentrations respectfully.

In all cases, robots that had developed in the negative environment showed a very "withdrawn" behaviour: the robot spent a significant portion of its time executing a behaviour similar to wall following. If the robot found a corner or an enclosed area, it would remain stationary in this location until other internal needs (e.g., the need to replenish energy) became more prioritarian. The reason for stopping in these enclosed areas was likely to be the fact that they were perceived as the safest location - as walls, which, detected on multiple sides, in a stressed state would have positive valence, effectively treating them as nests. As can be seen in figure 4, interaction with other areas of the environment was minimal due to the constant high level of the stress hormone, which suppressed the HPG-Axis effectively, preventing the emergence of a ratio between curiosity and stress that would modulate the ASM into investigating novel objects. In a few rare occasions that the robot did have a high enough level of curiosity hormone to facilitate and initiate interaction with novel objects, it quickly became over-stimulated and reverted to the previous withdrawn behaviour. Stress responses in

regards to interactions with novel objects not only tended to be more prevalent in these robots but also were significantly more severe and lasted on average 60 percent longer. Stress responses and hypersensitivity in regards to homeostatic deficits where also heightened in the robots that developed in a negative environment. Essentially, this meant that the robot would look to maintain homeostatic deficits at a higher level and if they started to drop, the robot would quickly enter it withdrawn behaviour. The implication of this is that, once the robot found an area of the environment with access to both resources, it would tend to stay in that general region and never really explore for new opportunities.

In contrast, the robots that had *developed in the positive environment* showed a much more outgoing behaviour, thoroughly exploring the entirety of the environment and interacting with a large range of the different novel objects. While this outgoing behaviour did lead to increased risks such as collisions or over stimulation, which caused the two high stress moments that can be seen in figure 4, the robot was able to recover fairly quickly. In addition, robots developed in a positive environment tended to have a greater tolerance to homeostatic deficits, which made them spend more time exploring and interacting with the environment.

When comparing the two robots developed under different environmental conditions, one additional significant difference can be observed in figure 4: the manner in which hormones were secreted, which tended towards quick large releases in the robot from the negative environment compared to smaller sustained releases from the robot with positive background. The quick bursts led to more unpredictable but responsive behaviour in regards to environmental stimuli. This could perhaps be compared to a highly responsive "flight-or-fight" system.

Experiments in the dynamic environment. In our second environment, dynamism was introduced by the presence of a human who interacted with the robot. Once again, the robots in the different runs had developed either in a positive or a negative environment. Due to the limited range of the robot's sensors, the range of "recognisable" interactions was relatively small.

As we could expect, the robots that had developed in a negative environment had a "timid demeanour" and tried to avoid any form of human interaction. However, gentle stroking motions along the IR-sensors could be used to initiate interaction by causing a rise in the concentration of the curiosity hormone. Interaction was primarily limited to this slow stroking as well as the robot exploring the human at its own pace. Any sudden movements or overzealous stroking would quickly lead to overstimulation of the robot and an attempt to withdraw. However, even with an ideal level of interaction, the hypersensitivity to homeostatic deficits meant the robot would only spend a maximum of around 30 seconds interacting before becoming more interested in procuring resources. It should be noted that robots that had developed such an aversive phenotype would try to withdraw immediately regardless of the interaction that the human tried to have. In addition, continuing to engage with any of these robots after they had attempted to withdraw led to an "aggressive" behaviour were the robot would attempt to push past or in a few case even turn to face the human and drive into them. We are still investigating the reason behind the emergence of this behaviour; however, we know that, for it to be triggered, the robot needs to have both an extremely high concentration of the stress hormone and a medium to high concentration of the curiosity hormone. Due to the inhibiting relation of the HPA-HPG axis, this concentration mix is a relatively rare phenomenon, and so far has only been achieved in the negative robot. These incidents represented the only time any of the robots made physical contact, excluding accidental collisions.

As we also expected, the robot that developed in a positive environment was much more tolerant of interaction with the human. Slow- to medium-speed stroking led to an initial positive response; after a period of interaction, faster stroking and sudden movements were tolerated and even sparked interest from the robot. To this extent, if an object such as a ball was thrown, the robot would go after it to investigate. Once the ball/object stopped and the robot had explored it as a normal novel object in the environment, the interest in the object would drop, often leading to the robot to return to the human in search of increased stimulation. It is worth noting that the robot did not know who or what had thrown the object and returned to the human purely because s/he is a large moving object and therefore had a high positive valence). As a comparison, an object thrown at a robot from the negative environment almost always led to the robot's withdrawal. An interesting aspect that emerged was the ability of the human to calm down the robot (i.e., to reduce its level of curiosity) by using low stimulation in the interaction.

Conclusion

In this paper we have presented and demonstrated a biologically plausible epigenetic mechanism that modulates the development of a wide range of value-laden survival and social behaviours, taking inspiration from a key component in the formation of motivations and behaviours: the HPA-HPG axis. As this epigenetic mechanism is dependent upon the internal state of the robot and its exposure to different environmental stimuli, the final phenotype of the robots reflects the conditions in which it developed. Our results show that a robot that developed in a negative environment spent a majority of its time thereafter trying to avoid any interaction with anything new or novel, preferring instead to simply stav in a safe location and maximise its homeostatic levels, providing a buffer to help protect itself from it perceived environmental dangers. In contrast, a robot developed in a positive environment spent a large portion of its time interacting with and exploring its surroundings. This included interaction with a human, for which the robot had not been programmed and that emerged as a consequence of the developmental history of the robot. Based upon our results in this paper, we argue that this model can provide a useful adaptive mechanism for autonomous robots to develop

behaviour that allows them to interact appropriately with different elements of a novel (physical and social) environment. In the future, we will look to expand this model by running experiments over longer periods of time and utilising the learning algorithm mentioned in the ASM section. This model will allow us to investigate the potential roles that hormones may play in modulating learning experiences.

References

- Arnold, Emotion and personality (1960) New York, NY, US: Columbia University Press.
- Avila-García, O. and Cañamero, L. (2004). Using hormonal feedback to modulate action selection in a competitive scenario. In: From Animals to Animats: Proceedings of the 8th International conference of Adaptive Behavior (SAB'04), p.p 243–252.
- Bechara A, Damasio H, Damasio AR (2000) Emotion, decision making and the orbitofrontal cortex. Cerebral Cortex 10:295-307.
- Buss K,A, Schumacher J, R, M, Dolski , I, Kalin N.H, Goldsmith H.H, Davidson R J (2003) Right frontal brain activity, cortisol, and withdrawal behavior in 6-month-old infantsBehavioral Neuroscience, 117, pp. 11–20
- Blanchard, A. and Cañamero, L. (2006). Developing Affect-Modulated Behaviors: Stability, Exploration, Exploitation, or Imitation? In F. Kaplan et al. (eds.), Proc. 6th International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems (EpiRob 6), 128: 17-24.
- Cañamero, L.D., Fredslund, J. (2000). How Does It Feel? Emotional Interaction with a Humanoid LEGO Robot. In K. Dautenhahn, ed., Socially Intelligent Agents: The Human in the Loop. AAAI 2000 Fall Symposium, pp. 23-28.
- Chelian, S. E., Oros, N., Zaldivar, A., Krichmar, J., & Bhattacharyya, R. (2012). "Model of the interactions between neuromodulators and pre-frontal cortex during a resource allocation task" in ICDL- EpiRob
- Daitzman, R. J., & Zuckerman, M. (1980). Disinhibitory sensation seeking and gonadal hormones. Personality and Individual Differences, 1, 103–110.
- Du Ruisseau, P., Y. TachS, P. Brazeau and R. Collu.(1979). Effects of chronic immobilization stress on pituitary hormone secretion, on hypothalamic factor levels, and on pituitary responsiveness of LHRH and TRH in female rats. Neuroendo- crinology 29:90
- French, R. and Cañamero, L. (2005). Introducing Neuromodulation to a Braitenberg Vehicle. In Proc. IEEE Intl. Conference on Robotics and Automation, "Robots get Closer to Humans" (ICRA 2005), pp. 4199-4204.
- Kalin, N.H. (1993). The neurobiology of fear. Scientific American, 268, 94-101
- Koolhass, J.M., van den Brink, T.H.C., Roozendaal, B, & Boorsma,F. (1990) Medial amygdala and aggressive behavior: Interactionbetween testosterone and vasopressin. Aggressive Behav., 16:223–229.

- Krichmar, J. L. (2012). "A biologically inspired action selection algorithm based on principles of neuromodulation" in The 2012 International Joint Conference on neural networks p.p 1-8
- LeDoux J, E Emotional memory systems in the brain (1993) Behavi. Brain Res., 58, pp. 69–79
- LeRoith, D., Shiloach, J., Roth, J., and Lesniak, M.A(1980). The evolutionary origins of vertebrate hormones: Insulin in unicellular organisms. Proc. Natl. Acad. Sci. USA 77:6184-6188
- Lones, J & Cañamero, L (2013), 'Epigenetic adaptation through hormone modulation in autonomous robots'. In IEEE ICDL-EPIROB 2013: The Third Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics
- Lones, J, Cañamero, L & Lewis, M (2013), 'Epigenetic adaptation in action selection environments with temporal dynamics'. in Advances in Artificial Life, ECAL 2013: 12th Conf on the Synthesis and Simulation of Living Systems. *MIT Press, pp. 505-512*.
- Lones, J, Cañamero, L & Lewis, M (2014), Hormonal modulation of interaction between autonomous agents In Press for The Forth Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics
- Liu D, Tannenbaum B, Caldji C, Francis D,Freedman A, et al. (1997). Maternal care, hippocampal glucocorticoid receptor gene expression and hypothalamic-pituitary-adrenal responses to stress. Science 277:1659–62
- Mazur, Allan, and Alan Booth. (1998). "Testosterone and Dominance in Men." Behavioral and Brain Sciences 21:353-63
- Mehta PH, Beer J. (2010) Neural mechanisms of the testosterone–aggression relation: The role of orbitofrontal cortex. Journal of Cognitive Neuroscience. (10):2357–2368
- McGowan, P. O., Sasaki, A., Dymov, S., LaBoonté, B., Turecki, G., Szyf, M., *et al.* (2009). Epigenetic regulation of the glucocorticoid receptor in human brain associates with childhood abuse. *Nature Neuroscience*, 12, 342–348.
- Montoya, E, Terburg, D, A Bos, P, and Honk, J (2012) Testosterone, cortisol, and serotonin as key regulators of social aggression: A review and theoretical perspective Motiv Emot. 2012 March; 36(1): p.g 65–73.
- Popma A, . Vermeiren R, C.A.M.L. Geluk, T. Rinne, W. van den Brink, D.L. Knol (2007). Cortisol moderates the relationship between testosterone and aggression in delinquent male adolescents Biol Psychiatry, 61 (3) pp. 405–411
- Reik ,W ,Dean, W, Walter, J Epigenetic Reprogramming in Mammalian Development (2001)Science 10 August 2001: 293 (5532), 1089-1093
- van Wingen G, Mattern C, Verkes RJ, Buitelaar J, Fernandez G (2010) Testosterone reduces amygdala-orbitofrontal cortex coupling. Psychoneuroendocrinology 35: 105–113.
- Zhang, T Y, Labonte, B, Wen, X, L, Turecki, G, Meaney, M J (2013)Epigenetic Mechanisms for the Early Environmental Regulation of Hippocampal Glucocorticoid Receptor Gene Expression in Rodents and Humans: Neuropsychopharmacology 38 1 pp 111 123

Idiotypic networks for evolutionary controllers in virtual creatures

Nicola Capodieci¹, Emma Hart² and Giacomo Cabri¹

¹University of Modena and Reggio Emilia ²Edinburgh Napier University

Abstract

We propose a novel method for evolving adaptive locomotive strategies for virtual limbless creatures that addresses both functional and non-functional requirements, respectively the ability to avoid obstacles and to minimise spent energy. We describe an approach inspired by artificial immune systems, based on a dual-layer idiotypic network that results in a completely decentralised controller. Starting from a system initialised with five non-adaptive locomotion strategies, we show that an adaptive controller can evolve that both minimises energy requirements and maximises distance covered when compared to the initial strategies.

Introduction

In Blumberg and Galyean (1995) a virtual creature is defined as an animate object, capable of goal-directed and time-varying behaviour, situated within a simulated environment with which it can interact. In this paper, we will focus on how to provide the means for a virtual creature to discover adaptive movement patterns; this will be accomplished by applying a previously introduced design methodology that merges artificial immune systems (AIS) and autonomic computing (Kephart and Chess (2003)). AIS takes inspiration from the biological immune system, in order to extract algorithms and methodologies for designing computational systems able to feature the same characteristics of the biological immune system, such as scalability, adaptivity, emerging cognition and decentralization. The evolutionary features able to provide adaptivity in the observed systems depends on which AIS related paradigm is used for solving a specified computational problem (De Castro and Timmis (2002)). In this instance, idiotypic networks as theorized by Cohen (2000a,b) have been used, due to their ability to show cognition. We are going to model our creature as a collection of independent (autonomic) units, with the aim of discovering new trajectories of movement and detect/avoid obstacles while optimizing the energy consumption of the virtual creature. This can be translated into a distributed autonomic computing related problem due to the composite morphologic nature of the creature itself and its requirement to Self-Adapt over time. Adaptation here is seen as the ability of the creature to combine known movement patterns with no adaptive ability in order to evolve new locomotion strategies able to keep the creature in motion. Although there is a wealth of literature related to the use of evolutionary methods for achieving control, our work differs in two key aspects. Firstly, it considers both functional and non-functional requirements of the creature, achieving movement while minimising energy. Secondly, our algorithm (that we name SelfEx) is completely distributed, in that every constituent unit of the virtual creature is able to evolve independently, sharing the minimum amount of information throughout the whole creature. Another difference compared to previous literature is the kind of network(s) involved: using an idiotypic network instead of a neural network (e.g. evolutionary morphologies as in Miconi and Channon (2005)) has the potential to further enrich the literature regarding evolutionary virtual creatures.

The paper is organized as follows: after presenting a review of related work, a description of the components of the virtual creature and its surrounding environment will be provided. The SelfEx approach is presented by detailing the modelling choices used. After the model is presented, all the details regarding the simulations and experiments performed and their related parameters will be given. The paper is concluded with discussions about the obtained results and related future research directions.

Related work

The vibrant field of Artificial life uses computer simulation to investigate evolution of behavioural and cognitive mechanisms in virtual creatures Sims (1994), potentially leading to advances in both biology (e.g. Palyanov et al. (2012)) and robotics (e.g. Černỳ and Kubalík (2013)). We restrict our review to work related to understanding the evolution of movement strategies that might ultimately be applied to the robotic field. A significant volume of work exists in the evolutionary computing literature, summarised by Prez-Moneo Surez and Rossi (2013) in relation to movement of limbless creatures. Typically, evolution evolves centralised

controllers in which performance is evaluated in terms of the evolved trajectory and ability to avoid obstacles but does not account for energy consumption of the movement, a relevant factor if the motion is to be transferred to real robots. Moreover, our approach (SelfEx, named after the Self-* property of a system to autonomously change its coordination pattern during run time execution of tasks (Cabri and Capodieci (2013)) uses a set of pre-coded movement strategies as baseline behaviours to evolve. This is in contrast with known evolutionary controllers in virtual creatures, since these latter ones rely on single atomic actions.

From the robotics perspective, a number of authors have advocated the use of immune-inspired control strategies, beginning with Ishiguro et al. (1995) (autonomous navigation for a single robot). This work was more recently extended by Whitbrook et al. (2010), in which the authors detailed how similar algorithms can be properly transferred in real robots. In Capodieci et al. (2013a), the authors proposed that ideas from autonomic computing could be combined with immune-inspiration to provided distributed control. An idiotypic network algorithm was proposed and applied to selecting movement strategies in swarm foraging task in Capodieci et al. (2013b). The model was formalised into a framework (Capodieci et al. (2014)) but only considered functional requirements.

The creature

The virtual creature used for our simulations is depicted in Figure 1 (left hand side). It is composed of 10 identical constituent units, each a perfect cube shape. Each unit is connected to its neighbouring unit(s) through a chain of universal joints, thus giving them complete freedom to rotate along the X axis, while rotations along the other axes are constrained by collisions with nearby units. The initial spacing between two units is set to one fifth of the length of the cube size. Trivially, the creature is simulated in a 3D environment in which the three axis have the orientation shown in Figure 1 (left hand side) and collision, friction and gravity forces are present. Each unit is completely independent — the only shared variable is an analogous to a biological clock that ensures synchronised adaptation of units. This is depicted in Figure 1 (right hand side) and is represented by a periodic square wave in which we can identify two distinctive phases, labelled as positive (P) and negative (N) phases. The use of a common shared clock is common in the field of virtual robotic creatures in attempting to imitate the oscillatory and periodic locomotion activity cycles of many existing animals (e.g. see Ijspeert (2008) for a survey on the existing methods for implementing Central Pattern Generators (CPGs)). The period of the clock can be adjusted according to T_{eval} .

Movements and related energy consumption

The functional objective of our creature is to keep moving and to discover new means of locomotion. In previous work, movement has been achieved by use of fixed functions, e.g. the serpenoid function as proposed by Hirose and Morishima (1990), applied at junctions on the body. However, this requires the use of an external and centralized controller. In the presented decentralised model, a generic notation for representing a movement pattern is to consider that during each clock phase, each unit independently applies a force from the centre of its mass with magnitude $M_g \leq 1$ with a direction described as a vector in which the unitary magnitude of the force can be distributed among the three axes: $(m_{qx}\mathbf{e_x} + m_{qy}\mathbf{e_y} + m_{qz}\mathbf{e_z})$, given $|m_{gx}|+|m_{gy}|+|m_{gz}|=M_g$ and where $\mathbf{e_x},\mathbf{e_y},\mathbf{e_z}$ are the canonical basis vectors in \mathbb{R}^3 . The direction of each force is always relative to the orientation of the unit. Movement of the creature therefore results from physical interactions between moving units; not all units are required to apply a force (and consume energy) for movement to occur: push or pulling behaviour of a unit can result from a force applied to a neighbouring unit. The energy spent by a single unit during a single time interval is calculated by summing the magnitudes of the applied force in both the positive and negative phase of the clock, therefore a single unit, during a time interval can show an energy consumption in the range [0, 2]. The total energy consumed (related the whole creature as indicated with EnC_{tot}) during a time interval is trivially calculated by summing the energy consumption experienced by each constituent unit during each phase of the clock.

Movement patterns

A series of 5 initial pre-coded movement patterns are used as starting points for evolution. A movement pattern is a description of how the whole creature moves and it is obtained by indicating magnitude and direction of the forces to be applied to each of its constituent unit during each phase of the clock. In the initial pre-coded movements, all forces magnitude are 1 and they are applied to one of the possible directions among $[\pm X, \pm Y, \pm Z]$. These movement patterns are loosely based on how limbless creatures move in space and are summarized in Table 1. In that table, for each unit (identified by a number from 0 to 9), the direction in which the force is applied is shown for each of the five pre-coded movement patterns (from Type1 to Type5). Individual units iterate the application of such forces as described in Table 1 during every time interval. Each initial pattern has an associated energy consumption that is fixed through the duration of an experiment. Moreover, the initial patterns do not enable obstacle avoidance or adaptation to external perturbations that cause unexpected rotations of the unit(s) and thus provide a baseline for evaluating whether these behaviours can emerge.

In Figure 2 the movement pattern labelled as Type1 is rep-

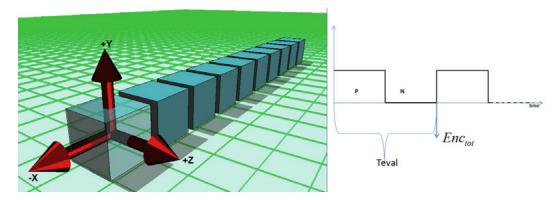


Figure 1: A representation of the virtual creature and its shared biological clock.

	Unit																				
	()	1	l l	2	2	3	3 4		5		6		7			8)	EnC_{tot}	
	P	N	P	N	P	N	P	N	P	N	P	N	P	N	P	N	P	N	P	N	1
Type1	+X	/	+X	/	+X	/	+Y	/	+Y	/	+Y	/	+Y	/	+Y	/	-X	+X	-X	+X	12
Type2	+X	+Y	+X	/	+X	/	+X	-Z	+Y	-Z	+Y	-Z	/	-Z	-X	/	-X	/	-X	+Y	15
Type3	+X	/	+X	/	+Y	/	+Y	/	+Y	/	+Y	/	/	-Z	/	-Z	/	-Z	/	-Z	10
Type4	+Y	+Z	+Y	+Z	+Y	+Z	+Y	+Z	/	/	/	/	+Z	/	+Z	/	+Z	/	+Z	/	12
Type5	-Z	/	-Z	/	-Z	/	-Z	/	-Z	/	-X	+Z	15								

Table 1: Table of initial movements patterns, Type1-Type5, showing force applied to each unit at each clock phase (P,N). EnC_{tot} : total consumed energy during T_{eval} . Example: in the movement pattern Type5, unit 0 applies a force of 1 magnitude over the negative Z axes in the positive (P) phase; no forces applied during the negative (N) phase.

resented and resembles the locomotion strategies adopted by caterpillars. Other movement patterns are based on rolling and crawling, similar to snakes or worms¹. It is important to stress that different movement patterns result in different outcomes both in terms of non-functional requirements (see the last column of Table 1, the total energy consumption EnC_{tot}) and in terms of functional requirements (calculated as the space distance the whole creature managed to travel during a time interval). Regarding the distance travelled by the creature, the performance varies according to the orientation of the constituent units, therefore any perturbation that causes the units to rotate from their original position can drastically change its performance and even prevent the creature to keep moving.

SelfEx: an idiotypic control model

Each constituent unit of the creature is seen as an autonomous component and has an associated $lymphnode\ l_{c_i}$. Each lymphnode is consisting of an internal network of interconnected antibodies. Additionally, an antibody in l_{c_i} may be connected by stimulatory or suppressive links to antibodies inside neighbouring lymphnodes $l_{c_{i+1}}$ and $l_{c_{i-1}}$, forming an $external\ network$. Antibody concentration varies over time according to the process outlined in Figure 3 and 4 and described in detail in the following sections. At the end of each time interval, the unit will select the highest concen-

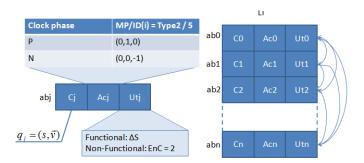


Figure 3: On the left side the model of a single antibody is shown. Example values are given to the action field (as verifiable from Table 1). The right part shows the connections among antibodies inside a single lymphnode.

trated antibody, so to extract from it a computational equivalent to an immunological response. Before describing the details of this process, it is important to show how the virtual antibodies are modelled.

Antibody Representation

An antibody is represented by a tuple $\langle condition\ C, action\ Ac, expected\ utility\ Ut \rangle$ and has an associated concentration (see Figure 3). This generic format was introduced in Capodieci et al. (2013b), and its implementation is application dependent. Here, the condition field is a quaternion in the form (s, \overrightarrow{v}) representing a possible orientation of the cube/unit in the three

¹All movement patterns are simulated for reference in a video resource located at https://vimeo.com/89119516

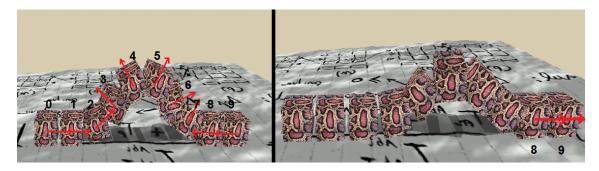


Figure 2: A graphical representation of the movement pattern "Type1". Left part: positive phase P of the clock, Right Part: negative phase N of the clock. Identifiers of the single units are also indicated in this picture.

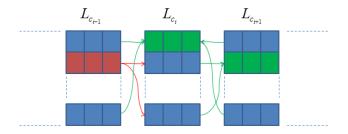


Figure 4: An example situation of how stimulation and suppression signals propagate in the external idiotypic network among three lymphnodes. Green antibodies resulted in positive feedback, while the red one obtained a negative feedback. Green arrows correspond to the increase of affinity by all the other unselected antibodies of neighbouring lymphnodes towards the green antibodies, while red arrows correspond to the increase of affinity by the red antibody towards all the other unselected antibodies.

possible directions of rotation. An *action* is a data-structure that represents a component identifier, the identifier of a movement pattern, and the magnitude and direction of the forces to be applied to the component at each clock phase. Finally, the *utility* field specifies two values corresponding to the expected value of the functional and non-functional measures. ΔS represents the functional utility and is simply the Euclidean distance between the starting position of the unit and its position at the end of a single time interval. The non-functional quantity is the energy consumption EnC (related to the single unit).

Initialisation and pre-experiments

Typically, initial antibodies populations in both AIS and evolutionary algorithms are created randomly. Instead, we seed the lymphnodes of each unit with a population of antibodies from information learned during a pre-experimental phase: each of the movement patterns described earlier are tested for a variable length of time (in terms of multiples of T_{eval}) to determine utility values. During this phase, external perturbations are applied randomly to the creature in order to

obtain a large interval of rotations as condition field of the antibodies. All antibodies are then initialised with concentration c_{init} . In the subsequent sensing phase, in each lymphnode l_{c_i} , the distance between the orientation specified by each antibody in the lymphnode (q) and the current orientation of the unit (q') is determined according to equation 1(a). Within each lymphnode lc_i , antibodies are ordered according to distance and assigned a fitness f_i according to equation 1(b), where ΔS_i and EnC_i are expected functional and non-functional utilities according to the utility field. Using a fitness proportionate selection method based on these fitness values, an antibody is chosen and its action applied to the relevant component (testing phase). It is important to notice that, since this very first step of the algorithm, each unit/lymphnode of the creature can choose actions related to different movement patterns, hence the search for alternative trajectories begins in the first time interval. The testing phase is followed by an evaluation phase in which the adaptation mechanisms take place.

(a)
$$d(q, q') = 2(1 - |q \cdot q'|)$$

(b) $f_i = \frac{\Delta S_i}{1 + EnC_i}$ (1)

Adaptation

This is the adaptation phase in which the original movement patterns evolve to optimise the functional and non-functional requirements of the creature. New antibodies are created that combine and/or adapt components of the 5 initial movement patterns into new strategies (see algorithm 1).

Affinity At the end of the evaluation phase, within each lymphnode, the currently active antibody ab_* (as the last selected antibody) discriminates between situations that lead to positive or negative feedback. Positive feedback is received whenever the obtained ΔS_o is greater or equal to the expected ΔS_{exp} stored in the currently selected antibody; negative feedback is received otherwise. Within each internal network, each antibody $ab \in l$ increases its affinity to ab_* if positive feedback is received. In case

(a)
$$\frac{\Delta c_{i}'}{\Delta t} = K_{p} \sum_{j=0}^{N} r_{j,i} c_{i} c_{j} - K_{n} \sum_{k=0}^{N} r_{i,k} c_{i} c_{k} + K_{D} \frac{d(q, q')}{1 + EnC_{i}} - K_{d} c_{i}$$
(b)
$$r_{i,j} = \frac{T_{ni} + T_{pj}}{T_{i,j}} (1 + |\Delta S_{o} - \Delta S_{exp}|)$$
(c)
$$\frac{\Delta c_{i}}{\Delta t} = K_{int} C_{i_{int}} + K_{ext} C_{i_{ext}}$$
(2)

of a negative feedback, the previously selected antibody ab_* will increase its affinity towards the other unselected antibodies $ab \in l$. Affinities are used for calculating the variation of concentration for the antibodies, as shown in equation 2(a). Affinity $r_{i,j}$ between generic antibodies iand j are calculated according to equation 2(b). The same mechanisms are also applied to the external network, as shown in the example situation depicted in Figure 4. The purpose of the external network is to share experiences throughout the creature: if, for instance, lymphnode l_{c_i} obtained a negative feedback, the reason for it could be ascribed to the neighbouring units that did not push/pulled in the right direction; vice-versa, if the same lymphnode obtained a positive feedback, the other neighbouring units should be aware of that, even if they experienced a negative The SelfEx algorithm should be able to find a balance between these situations.

Equation 2(b) is borrowed directly from Ishiguro et al. (1995) and forces selection of the same antibody in a subsequent similar situation (in the case of positive feedback) and vice-versa. T_{pj} corresponds to the total number of evaluation periods over which ab_j received positive feedback and T_{ni} are the total number of times ab_i received a penalty; $T_{i,j}$ is the number of times both antibodies i and j have been activated. In contrast to Ishiguro et al. (1995) we multiply the quantity representing the ratio of positive to negative feedback by a term proportional to the difference between obtained and expected distance travelled to represent functional utility.

Dynamics The *concentration* c_i of an antibody i determines its ability to compete against other antibodies to be able to execute its action and is dependent on its affinity with other antibodies. We apply an equation very similar to the differential equation suggested by Ishiguro et al. (1995) to modify concentration, given in equation 2(a). As visible in equation 2(a), we can identify four components and each component is controlled by a constant value. The first term represents *stimulation* of an antibody, the second term *suppression*, the third term the *stimulation* of the antibody from the environment (proportional by the distance between conditions and inversely proportional by the energy consumption measure), and the final term is a *decay* term represent-

ing the tendency of antibodies to die if not stimulated. In order to calculate the concentration of an antibody due to its *internal* network, $C_{i_{int}}$, the equation is applied over the N_i antibodies in the lymphnode of antibody i. The concentration of the i due its *external* network $C_{i_{ext}}$ is calculated by applying the equation over the N_e antibodies in its external network. Finally, its total concentration its calculated by weighting the contributions from the internal and external network as in (c) where K_{int} and K_{ext} are constants. The final value of concentration is then stabilized through a squashing function as defined in Ishiguro et al. (1995).

Meta-dynamics Meta-dynamic processes create new antibodies which can become integrated into the network. In contrast to previous meta-dynamic models (e.g. Ishiguro et al. (1995)), in our SelfEx model, the creation of new antibodies is not a continuous process but is triggered by stagnation in the network. Specifically, mutation is triggered whenever more than one antibody converges to the same concentration, a condition that (according to initial experiments) occurs when the creature becomes stuck over a prolonged period of time. The α mutation operator, triggered when two or more antibodies reach the same concentration level, creates a new antibody in which the condition field is set to the current unit orientation and all other fields are averaged among all the antibodies that share the same concentration value. To better understand what averaging the action field of an antibody means, we can consider an example situation in which two antibodies (i and j) share the same highest concentration value: let us suppose that the action during the positive phase of i is (0,0,-1) while in j is (0,1,0) then the new antibody k will feature an action during the positive phase of (0,0.5,-0.5). As a result, new antibodies enable the unit to move though composite directions, hence providing the single unit with the basis to create new movement patterns.

A second mutation operator refines new antibodies in order to generate new obstacle avoidance strategies. Mutation β occurs according to a probability P_{β} as described in equation 3. M is the current number of antibodies and it increases whenever an α mutation occurs, while N is the initial number of antibodies. K_a is a constant that regulates how often this kind of mutation can occur. As soon as M-N>0,

there is a non-zero probability that a new antibody will be created featuring the same condition and expected utility field as the last selected antibody. The action field is calculated by detecting the dominant direction of the last antibody and then inverting its direction. For example, if the positive phase of the last selected antibody i featured an action represented as (0.8,0.2,0), then the newly generated antibody k will feature an action in the positive phase represented as (-0.8,0,0). Once a β mutation occurs, N is set to M and all the affinity and concentration values are reset to their initial values.

$$P_{\beta} = \frac{M - N}{K_a} \tag{3}$$

Experiments and Results

After the pre-experimental phase, the same initial values of inter-antibodies affinities and concentrations are assigned to all the antibodies. The mechanism used in the experiment is shown in algorithm 1.

```
Algorithm 1: mechanisms of adaptation
```

```
Data: Initial population of antibodies after pre-experiments
Executed by: each lymphocyte i;
Sensing: current orientation \rightarrow Select antibody \rightarrow action;
TEST: while (t < t_{eval}) do
    test the selected action;
end
performance = evaluate(\Delta S, Enc);
updated affinities and concentrations;
if \alphaMutationCondition is true then
    applyMutation\alpha;
    if \betaMutationCondition is true then
        applyMutation\beta;
    end
    add newly generated antibodies to list of antibodies;
    select newly generated antibody;
end
else
    Select highest concentrated antibody;
end
goto: TEST;
```

The algorithm is tested in an arena represented by a room that is confined between 4 walls. The algorithm is implemented in Javatm (jdk v. 1.7.0_45) using *JOGL* (http://jogamp.org/jogl/www/) for visualization purposes and *jinngine* as physics engine (Silcowitz-Hansen (2010)). After the pre-experimental phase (performed just once before the experiments), each unit in the creature is initialised with a population of 50 antibodies; each antibody is then set to the initial concentration $c_i = 0.5$. Parameters that were fixed during the experiments were $K_a = 3$, $K_d = 0.25$, $T_{eval} = 2s$ while all the other parameters undergo a process of search in order to find their best combinations resulting in settings of $K_p = 1$, $K_n = 0.85$, $K_D = 0.5$, $K_{int} = 0.45$

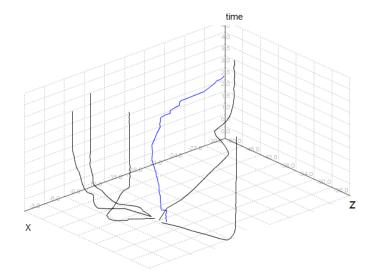


Figure 5: 50 time interval experiment in which all the trajectories caused by the pre-coded movement patterns (in black) are compared to SelfEx (in blue).

and $K_{ext} = 0.55$. Experiments are repeated 10 times with the total energy consumed and average distance covered (per single time interval) over varying time intervals; both these indicators are averaged throughout the whole creature. Experimental goals are to: (a) investigate whether pre-coded movement patterns can be combined in order to show different movement trajectories; (b) optimise distance travelled (c) optimise energy consumption.

Indicative results showing trajectories over a single run are depicted in Figure 5 and Figure 6. The y axis indicates movement over time, thus stationary behaviour is observed whenever there is no change in the XZ plane.

In Figure 5 shows that the evolved SelfEx trajectory differs from the single pre-coded movement patterns in that initially it makes slow progress, but once movement is established it continues throughout the experiment in a different trajectory.

In Figure 6 obstacle avoidance in addition to continuous movement is clear, as the plot shows how the creature manage to step back from walls.

In Figure 7 total energy consumption over time is shown, compared to the least costly single movement pattern (Type3) and the most costly (Type2, Type5). Over 10 runs, the average energy consumption is 10.11 (standard deviation $\sigma=0.128$), while the average distance covered per single time interval is $avg\Delta S=0.38$ ($\sigma=0.129$). The comparison with the 5 pre-coded movement patterns is shown in Table 2^2 . Our SelfEx algorithm was able to discover new trajectories by combining *pieces* of pre-coded move-

²Do note that the outcomes of the pre-coded patterns are deterministic.

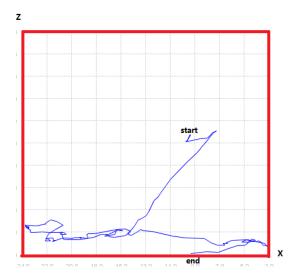


Figure 6: SelfEx trajectory over the XZ plane during 190 time intervals; walls are indicated in red.

MP	EnC_{tot}	$avg\Delta S$
Type1	12	0.26
Type2	15	0.21
Type3	10	0.21
Type4	12	0.33
Type5	15	0.25
SelfEx	10.11	0.38

Table 2: Average perfomance

ment patterns starting from the very first time interval of the experiment. This is an implication of the fact that each unit/lymphnode selects actions related to different movement patterns, thus creating new locomotion strategies even before starting generating new antibodies. The generation of new antibodies occurs whenever mixing the pre-coded strategies still does not cause movement to the creature; moreover the fitness function used during the sensing phase (equation 1(b)) and the third term of eq. 2 (a) forces the unit to constantly take into account its energy consumption, since the increment of the concentration value is inversely proportional to the energy cost of the movement: even if an antibody caused a positive feedback, the increase of the antibody concentration will not be substantial if the consumed energy is high; as an implication of this, less energy consuming actions are always selected.

Conclusions and future work

The paper has presented a modified version of an idiotypic network algorithm to evolve movement in a virtual creature. The algorithm was inspired by early work in robotic control by Ishiguro et al. (1995) and extends previous work in which a modified version of this algorithm was used to

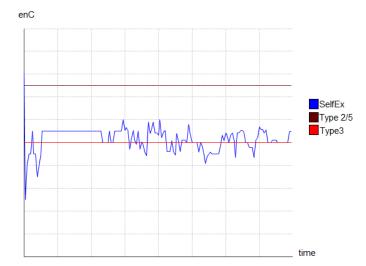


Figure 7: Energy consumption of SelfEx compared to Type1 (low energy), and Type3/Type5 (high energy).

select coordination patterns in swarm robotic applications (Capodieci et al. (2013b)). The presented control system is fully distributed, adaptive, and accounts for both functional and non-functional requirements. By only sharing a biological clock and by relying on a dual idiotypic network, the creature minimizes energy consumption and at the same time, discovers new movement trajectories and obstacles avoidance behaviours. Its novelty lies in the use of a 2-layer network, where antibodies interact in both internal and external networks, trading off unit-control against shared behaviours. Furthermore, the algorithm generates novel locomotion patterns in two ways: by combining fragments of existing patterns and by generating new patterns by mutating existing ones. As for future work, applying the same dual immune network for creating evolutionary morphologies for the creature is definitely an interesting development to be exploited in the near future.

Acknowledgement

The work is partially supported by the ASCENS project (EU FP7-FET, Contract No. 257414)

References

Blumberg, B. M. and Galyean, T. A. (1995). Multi-level direction of autonomous creatures for real-time virtual environments. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 47–54. ACM.

Cabri, G. and Capodieci, N. (2013). Runtime change of collaboration patterns in autonomic systems: Motivations and perspectives. In Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on, pages 1038–1043.

- Capodieci, N., Hart, E., and Cabri, G. (2013a). Designing self-aware adaptive systems: from autonomic computing to cognitive immune networks. In Proceedings of the 3rd Workshop on Challenges for Achieving Self-Awareness in Autonomic Systems, Philadelphia, USA.
- Capodieci, N., Hart, E., and Cabri, G. (2013b). An immune network approach for self-adaptive ensembles of autonomic components: a case study in swarm robotics. In *Advances in Artificial Life, ECAL*, volume 12, pages 864–871.
- Capodieci, N., Hart, E., and Cabri, G. (2014). Artificial immune system in the context of autonomic computing: integrating design paradigms. *Proceedings of Genetic and Evolutionary Computation Conference, GECCO 2014*, (-):to appear.
- Černý, J. and Kubalík, J. (2013). Co-evolutionary approach to design of robotic gait. In *Applications of Evolutionary Computation*, pages 550–559. Springer.
- Cohen, I. R. (2000a). Discrimination and dialogue in the immune system. In *Seminars in Immunology*, volume 12, pages 215– 219. Elsevier.
- Cohen, I. R. (2000b). *Tending Adam's Garden: evolving the cognitive immune self.* Academic Press.
- De Castro, L. N. and Timmis, J. (2002). *Artificial immune systems:* a new computational intelligence approach. Springer.
- Hirose, S. and Morishima, A. (1990). Design and control of a mobile robot with an articulated body. *The International Journal of Robotics Research*, 9(2):99–114.
- Ijspeert, A. J. (2008). Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, 21(4):642–653.
- Ishiguro, A., Watanabe, R., and Uchikawa, Y. (1995). An immunological approach to dynamic behavior control for autonomous mobile robots. In *Intelligent Robots and Systems 95. Human Robot Interaction and Cooperative Robots'*, Proceedings. 1995 IEEE/RSJ. IEEE.
- Kephart, J. and Chess, D. (2003). The vision of autonomic computing. *Computer*, 36(1):41–50.
- Miconi, T. and Channon, A. (2005). A virtual creatures model for studies in artificial evolution. In *Evolutionary Computation*, 2005. The 2005 IEEE Congress on, volume 1, pages 565– 572. IEEE.
- Palyanov, A., Khayrulin, S., Larson, S. D., and Dibert, A. (2012). Towards a virtual c. elegans: A framework for simulation and visualization of the neuromuscular system in a 3d physical environment. *In silico biology*, 11(3):137–147.
- Prez-Moneo Surez, D. and Rossi, C. (2013). A comparison between different encoding strategies for snake-like robot controllers. In *Applications of Evolutionary Computation*, volume 7835 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg.
- Silcowitz-Hansen, M. (2008-2010). Jinngine, a physics engine written in java.
- Sims, K. (1994). Evolving 3d morphology and behavior by competition. *Artificial life*, 1(4):353–372.

Whitbrook, A. M., Aickelin, U., and Garibaldi, J. M. (2010). Real-world transfer of evolved artificial immune system behaviours between small and large scale robotic platforms. *Evolutionary Intelligence*, 3(3-4):123–136.

Hold the Spot: Evolution of Generalized Station Keeping for an Aquatic Robot

Jared M. Moore and Anthony J. Clark

Michigan State University, East Lansing, MI, USA 48824 moore112@msu.edu

Abstract

In this paper, we present a strategy to evolve neurocontrollers in aquatic robots capable of generalized station keeping, that is, maintaining a position in the presence of various water flows. Evolved behaviors exhibit a variety of complex fin/flipper movements that enable the robot to react and move against changing flows. Moreover, results indicate that some sensor modalities are beneficial when the robot is placed in novel environments, though little used during the evolutionary process.

Introduction. Increasingly, aquatic robotic systems are being deployed to assist humans in challenging tasks (Tan et al., 2006). Although many systems rely on control from a human, autonomy allows robots to act independently in hostile or remote environments. Station keeping, also known as station holding, involves maintaining a position against external fluid flows, and is exhibited by many biological fish (Arnold, 1974). It is also of interest for aquatic robot sensor platforms that need to remain stationary while gathering data. In such cases, the autonomous control system needs to be able to respond to changing flows.

We previously examined the evolution of station keeping behaviors for individuals facing a single flow during their evaluation and evolutionary periods (Moore et al., 2013). Although successful in this task, individuals failed to maintain station when facing novel flows (i.e., flows that were not encountered during the evolutionary process). In this work, we focus on the evolution of *generalized* station keeping behaviors capable of handling multiple distinct flows. We investigate approaches to this problem through two separate experimental setups that utilize multiple flows during the evolutionary fitness evaluation. Evolved individuals are then evaluated in both previously seen and novel flows.

Results indicate that evolved individuals are able to hold station against flows encountered within and outside of the evolutionary process, exhibiting some generalized behaviors. Furthermore, additional sensor modalities increase an individual's ability to generalize to new flow conditions, even though they do not appear to be beneficial in environments encountered during evolution. This work provides an approach to developing a generalized control strategy for dynamic environmental conditions, and provides insight into the impact of sensory information for evolved neural controllers.

Methods. The simulated robot in this study emulates the form and function of a physical device, seen in Figure 1a. Pectoral flippers are capable of continuous 360° range of motion, while the caudal fin is limited to a $\pm 90^{\circ}$ symmetric range of motion. Sensory information includes inertial data (i.e., linear and angular acceleration at the robot's center of mass) and the previous update's motor commands. Information about the actual state of the motors is not used as most small continuous rotation servo motors do not provide this data. In two of the treatments, we include additional sensory information including the robot's pitch, roll, and yaw, along with the xyz flow information.



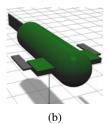


Figure 1: (a) Physical robot with 3D printed components. (b) Simulated agent derived from physical robot.

The Open Dynamics Engine (Smith, 2013), a 3D rigid-body physics library, is employed as the simulation environment. We extended ODE with a fluid dynamics model, discussed in (Moore et al., 2013), based on hydrodynamic drag adapted from (Wang et al., 2011) and (Sims, 1994). Neural controllers are evolved with the NEAT algorithm (Stanley and Miikkulainen, 2002). Individuals are evaluated based on their distance from the station point at 250ms intervals over a 60s evaluation period. The closer an individual is to the station point, the higher its fitness for that interval.

Experiments and Results. Videos of selected results are available at the following links:

```
Video 1: http://youtu.be/MO-ueGP3eG0
Video 2: http://youtu.be/HXUwr6WEdLU
Video 3: http://youtu.be/HhMTkf0FUfY
Video 4: http://youtu.be/05oSypwWhyo
Video 5: http://youtu.be/kL-KRjXL0kQ
```

Treatments 1 and 2 are conducted in an environment with a gradually changing side-to-side flow. At the start of an individual's evaluation, the flow originates from the front of the robot. The direction of the flow moves to one side, reaching its maximum angular offset of 63.4° at 15s. Halfway through the simulation (30s), the flow returns to the center, then moves to the opposite side with respect to the robot's initial orientation. A second environmental setup in Treatments 3 and 4 focus on holding station against a flow, then moving and stopping again at another flow. Five flows are possible: -45° , -22.5° , 0° , 22.5° , and 45° , of which two are randomly selected for an individual evaluation. Agents must be able to detect changes in the flow, or lack thereof. A total of nine possible flow combinations are possible during evolution.

Evolutionary results indicate that Treatment 1 (lacking additional sensors) slightly outperforms Treatment 2 during evolution. In Treatments 3 and 4, a reduced sensory capacity also leads to higher fitnesses in environments seen during the evolutionary process. However, in all treatments, evolved agents are able to maintain station effectively in the evolutionary environments.

The focus of this study, the evolution of generalized behavior, assesses individuals based on how they perform in novel flow conditions. Contrary to the evolutionary results, individuals evolved with additional sensory input achieve the best station keeping in novel environments. We hypothesize that the additional sensory information may be extra noise for an evolved ANN in a familiar environment (i.e. one seen during the evolutionary process), but is important information when encountering environments not previously seen.

We test evolved individuals in an environment containing both a change in direction and magnitude of the flow. Here, the flow begins at twice the magnitude encountered during evolution and is strong enough that individuals are physically unable to generate enough thrust to hold station. The flow is reduced after the first 20s of a 60s simulation, allowing agents to swim back towards the station point. Two individuals from Treatment 1 and 4 are shown in Video 5 with an individual from Treatment 4 depicted in Figure 2. Here, sensors are again beneficial in novel environments as the best performing individuals come from Treatment 4, which has the additional sensory information. Behaviors exhibited in this test indicate that these individuals are not evolved to swim against the flows at a steady rate, but can return to station when displaced by stronger, previously unseen flows.

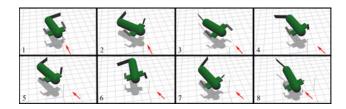


Figure 2: An evolved individual from Treatment 4 exhibits station keeping behavior by swimming back towards the origin after initially being pushed away from the station by a strong flow.

Discussion Unlike previous work (Moore et al., 2013), these treatments were effective in eliciting generalized station keeping. Surprisingly, the additional sensory information does not appear to increase the ability of individuals to hold station in environments encountered during the evolutionary process. However, when placed in previously unseen environments, individuals with extra sensory information are more effective than those relying on a limited set of sensors. This suggests that additional sensory information can be beneficial in unforeseen conditions. Future work includes determining why the additional information is beneficial, and pursuing the evolution of generalized controllers for different tasks.

References

Arnold, G. P. (1974). Rheotropism in fishes. *Biological Reviews*, 49(4):515–576.

Moore, J. M., Clark, A. J., and McKinley, P. K. (2013). Evolution of station keeping as a response to flows in an aquatic robot. In *Proceedings of the 2013 ACM Genetic and Evolutionary Computing Conference*, pages 239–246, Amsterdam, Netherlands. ACM.

Sims, K. (1994). Evolving virtual creatures. In *Proceedings* of the 21st Annual Conference on Computer Graphics and Interactive Techniques, pages 15–22.

Smith, R. (2013). Open Dynamics Engine, http://www.ode.org/.

Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.

Tan, X., Kim, D., Usher, N., Laboy, D., Jackson, J., Kapetanovic, A., Rapai, J., Sabadus, B., and Zhou, X. (2006). An autonomous robotic fish for mobile sensing. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5424 –5429, Beijing, China.

Wang, J., Alequin-Ramos, F., and Tan, X. (2011). Dynamic modeling of robotic fish and its experimental validation. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 588 –594, San Francisco, California, USA.

Real-time Hebbian Learning from Autoencoder Features for Control Tasks

Justin K. Pugh¹, Andrea Soltoggio², and Kenneth O. Stanley¹

Dept. of EECS (Computer Science Division), University of Central Florida, Orlando, FL 32816 USA 2Computer Science Department, Loughborough University, Loughborough LE11 3TU, UK jpugh@eecs.ucf.edu, a.soltoggio@lboro.ac.uk, kstanley@eecs.ucf.edu

Abstract

Neural plasticity and in particular Hebbian learning play an important role in many research areas related to artficial life. By allowing artificial neural networks (ANNs) to adjust their weights in real time, Hebbian ANNs can adapt over their lifetime. However, even as researchers improve and extend Hebbian learning, a fundamental limitation of such systems is that they learn correlations between preexisting static features and network outputs. A Hebbian ANN could in principle achieve significantly more if it could accumulate new features over its lifetime from which to learn correlations. Interestingly, autoencoders, which have recently gained prominence in deep learning, are themselves in effect a kind of feature accumulator that extract meaningful features from their inputs. The insight in this paper is that if an autoencoder is connected to a Hebbian learning layer, then the resulting Realtime Autoencoder-Augmented Hebbian Network (RAAHN) can actually learn new features (with the autoencoder) while simultaneously learning control policies from those new features (with the Hebbian layer) in real time as an agent experiences its environment. In this paper, the RAAHN is shown in a simulated robot maze navigation experiment to enable a controller to learn the perfect navigation strategy significantly more often than several Hebbian-based variant approaches that lack the autoencoder. In the long run, this approach opens up the intriguing possibility of real-time deep learning for control.

Introduction

As a medium for adaptation and learning, neural plasticity has long captivated artificial life and related fields (Baxter, 1992; Floreano and Urzelai, 2000; Niv et al., 2002; Soltoggio et al., 2008, 2007; Soltoggio and Jones, 2009; Soltoggio and Stanley, 2012; Risi and Stanley, 2010; Risi et al., 2011; Risi and Stanley, 2012; Stanley et al., 2003; Coleman and Blair, 2012). Much of this body of research focuses on Hebbian-inspired rules that change the weights of connections in proportion to the correlation of source and target neuron activations (Hebb, 1949). The simplicity of such Hebbian-inspired rules makes them easy and straightforward to integrate into larger systems and experiments, such as investigations into the evolution of plastic neural networks (Floreano and Urzelai, 2000; Soltoggio et al., 2008; Risi et al., 2011). Thus they have enabled inquiry into such diverse problems as task switching

(Floreano and Urzelai, 2000), neuromodulation (Soltoggio et al., 2008), the evolution of memory (Risi et al., 2011), and reward-mediated learning (Soltoggio and Stanley, 2012).

However, while Hebbian rules naturally facilitate learning correlations between actions and static features of the world, their application in particular to control tasks that require learning *new features* in real time is more complicated. While some models in neural computation in fact do enable low-level feature learning by placing Hebbian neurons in large topographic maps with lateral inhibition (Bednar and Miikkulainen, 2003), such low-level cortical models generally require prohibitive computational resources to integrate into real-time control tasks or especially into evolutionary experiments that require evaluating numerous separate individuals. Thus there is a need for a convenient and reliable feature generator that can accumulate features from which Hebbian neurons combined with neuromodulation (Soltoggio et al., 2008) can learn behaviors in real time.

Interestingly, such a feature-generating system already exists and in fact has become quite popular through the rise of deep learning (Bengio et al., 2007; Hinton et al., 2006; Le et al., 2012; Marc' Aurelio et al., 2007): the autoencoder (Hinton and Zemel, 1994; Bourlard and Kamp, 1988). Autoencoders, which can be trained through a variety of algorithms from Restricted Boltzmann Machines (RBMs) (Hinton et al., 2006) to more conventional stochastic gradient descent (Le et al., 2012) (e.g. similar to backpropagation; Rumelhart et al. 1986), aim simply to output the same pattern as they input. By training them to mimic their inputs, they are forced to learn key features in their hidden layer that efficiently encode such inputs. In this way, they accumulate such key features as they are exposed to more inputs. However, in deep learning autoencoders are usually trained for classification tasks through an unsupervised pre-training phase and in fact recent results have raised doubts on their necessity for such tasks anyway (Cireşan et al., 2012). The idea in this paper is that in fact autoencoders can instead be put to good use as feature accumulators that work synergistically with neuromodulated Hebbian connections that learn from the accumulating features in real time, as an autonomous

agent acts in the world. The resulting structure is the *Real-time Autoencoder-Augmented Hebbian Network* (RAAHN), a novel algorithm for learning control policies through rewards and penalties in real time.

In short, the key idea behind the RAAHN is that a missing ingredient that can reinvigorate the field of Hebbian learning is the ability of an autonomous agent to learn new features as it experiences the world. Those new features are then simultaneously the inputs to a neuromodulated Hebbian layer that learns to control the agent based on the accumulating features. In effect, the RAAHN realizes the philosophy that real-time reward-modulated learning cannot achieve its full potential unless the agent is simultaneously and continually learning to reinterpret and re-categorize its world. Introducing the RAAHN thereby creates the opportunity to build and study such systems concretely.

The experiment in this paper is intended as a proof of concept designed to demonstrate that it is indeed possible to learn autoencoded (and hence unsupervised) features at the same time as Hebbian connections are dynamically adapting based both on correlations with the improving feature set and neuromodulatory reward signals. In the experiment, simulated robots with two kinds of sensors (one to see the walls and the other to see a non-uniform distribution of "crumbs" on the floor) are guided through a maze to show them the optimal path, after which they are released to navigate on their own. However, supervised learning does not take place in the conventional sense during this guided period. Instead, both the autoencoder and the Hebbian connections are adjusting in real time without supervisory feedback to the autoencoder. Furthermore, to isolate the advantage of the autoencoder, two other variants are attempted - in one the Hebbian connections learn instead from the raw inputs and in the other the Hebbian connections learn from a set of random features (e.g. somewhat like an extreme learning machine; Huang and Wang 2011).

The main result is that only the full RAAHN learns the optimal path through the maze in more than a trivial percentage of runs, showing not only that it is possible to train an autoencoder and Hebbian connections simultaneously, but that in fact the autoencoder component can be essential for incorporating the most important features of the world.

While the RAAHN could be viewed in the context of reinforcement learning (RL), it is important to note that the approach is rather aimed at issues outside typical RL. In particular, the RAAHN can be viewed as a platform for later integrating more advanced and realistic Hebbian learning regimes, e.g. with distal rewards (Soltoggio et al., 2013) to demonstrate more convincingly their full potential.

In effect, the RAAHN is a new way to think about plasticity that goes beyond the proof of concept in this paper. It is among the first methods to suggest the potential for *realtime deep learning for control*. In that way it opens up a large and rich research area for which this paper represents

an initial step. If Hebbian connections can be trained from a dynamically adjusting autoencoder, then perhaps one day they will learn in real time from deepening stacked autoencoders or within complex evolved networks that incorporate autoencoders as a basic element. While much remains to be explored, the initial study here thereby hints at what might be possible in the future.

Background

This section reviews Hebbian learning in artificial neural networks (ANNs) and the application of autoencoders in deep learning.

Hebbian ANNs

The plain Hebbian plasticity rule is among the simplest for training ANNs:

$$\Delta w_i = \eta x_i y,\tag{1}$$

where w_i is the weight of the connection between two neurons with activation levels x_i and y, and η is the learning rate. As an entirely local learning rule, it is appealing both for its simplicity and biological plausibility. For this reason, many researchers have sought to uncover the full extent of functionality attainable by ANNs only of Hebbian rules.

As researchers have gained insight into such networks, they have also found ways to increase the rule's sophistication by elaborating on its central theme of strengthening through correlated firing (e.g. Oja 1982; Bienenstock et al. 1982). Researchers also began to evolve such ANNs in the hope of achieving more brain-like functionalities by producing networks that change over their lifetime (Floreano and Urzelai, 2000; Niv et al., 2002; Risi and Stanley, 2010; Risi et al., 2011; Stanley et al., 2003).

One especially important ingredient for Hebbian ANNs is neuromodulation, which in effect allows Hebbian connections to respond to rewards and penalties. Neuromodulation enables the increase, decrease, or reversal of Hebbian plasticity according to feedback from the environment. Models augmented with neuromodulation have been shown to implement a variety of typical features of animal operant learning such as reinforcement of rewarding actions, extinction of unproductive actions, and behavior reversal (Soltoggio and Stanley, 2012; Soltoggio et al., 2013). The combination of Hebbian ANNs with neuromodulatory signals in recent years has especially inspired neuroevolution and artificial life researchers by opening up the possibility of evolving ANNs that can learn from a sequence of rewards over their lifetime (Soltoggio et al., 2008, 2007; Soltoggio and Jones, 2009; Soltoggio and Stanley, 2012; Risi and Stanley, 2012; Coleman and Blair, 2012).

However, one limitation of these cited studies is that the inputs are generally heavily pre-processed to provide meaningful and useful feature to the neural substrate that performs Hebbian plasticity. A natural question is whether such useful features can in principle emerge in real-time during the

agent's lifetime, and in combination with the associative, reward-driven learning provided by Hebbian plasticity. As this paper argues, the autoencoder, reviewed next, is an appealing candidate for playing such a role.

Autoencoders in Deep Learning

The idea behind the autoencoder is to train a network with at least one hidden layer to reconstruct its inputs. The auto encoder can be described as a function f that encodes a feature vector x (i.e. the inputs) as a set of hidden features h = f(x). A second function q then decodes the hidden features h (typically a hidden layer within an ANN) into a reconstruction r = g(h) (Bengio et al., 2013). The hope of course is that once trained, r will be as close as possible to x for any input x. While many possible autoencoder models exist, the parameters of the autoencoder (which can be represented as weights in an ANN) are often the same for the encoder and decoder, which means in effect that the weights are bidirectional (Bengio et al., 2013). The main property of the autoencoder that makes it interesting is that by forcing it to learn hidden features h that can reconstruct input instances, under the right circumstances the features of h are forced to encode key features of the input domain. For example, edge detectors might arise in h for encoding images (Hinton et al., 2006).

Autoencoders began to gain in popularity considerably after researchers observed that they can help to train deep networks (i.e. ANNs of many hidden layers) through a pretraining phase in which a stack of autoencoders is trained in sequence, each one from the previous (Bengio et al., 2007; Hinton et al., 2006; Le et al., 2012; Marc'Aurelio et al., 2007), leading to a hierarchy of increasingly high-level features. Because it was thought that backpropagation struggles to train networks of many layers directly, pre-training a stack of such autoencoders and then later completing training through e.g. backpropagation was seen as an important solution to training deep networks. Although later results have suggested that in fact such pre-training is not always needed (Cireşan et al., 2010) (especially in the presence of an abundance of labeled data), it remains a compelling demonstration of unsupervised feature accumulation and remains important for training in the absence of ample labeled data (Bengio et al., 2013). In any case, typically the main application of such deep networks is in classification problems like handwriting recognition.

Another appeal of autoencoders is that there are many ways to train them and many tricks to encourage them to produce meaningful features (Ranzato et al., 2006; Le et al., 2012). While RBMs (a kind of probabilistic model) can play a similar role to autoencoders, classic autoencoders in deep learning are generally trained through some form of stochastic gradient descent (Le et al., 2012; Bengio et al., 2007) (like backpropagation), as is the case in this paper. However, the important issue in the present investigation

is not the particular details of the autoencoder; in fact an advantage of the RAAHN formulation is that any autoencoder can be plugged into the RAAHN. Thus as autoencoders are refined and improved, RAAHNs naturally benefit from such improvements and refinements. It is also possible that the real-time context of RAAHNs will provoke more attention in the future to identifying autoencoder formulations most suited to real time.

Approach: Real-time Autoencoder-Augmented Hebbian Network

The Real-time Autoencoder-Augmented Hebbian Network (RAAHN) approach introduced in this paper consists of two distinct components: the autoencoder and the Hebbian learner. The simplest implementation consists of an ANN with a single hidden layer; connections from the inputs to the hidden layer are trained as an autoencoder and connections from the hidden layer to the outputs are trained with a Hebbian rule. Thus the hidden layer represents a set of features extracted from the inputs that a Hebbian rule learns to correlate to the outputs to form an effective control policy.

Both of these components can be implemented in a number of different ways. The particular implementation described in this section, which is tested later in the experiment, serves as a proof of concept. It is designed accordingly to be as simple as possible.

Autoencoder Component

The autoencoder in this experiment is a heuristic approximation of the conventional autoencoder (Bengio et al., 2013) that was chosen for simplicity and ease of implementation. It is important to note that it suffices for the purpose of this experiment because it converges to within 5% of the optimal reconstruction in every run of the experiment in this paper. This consistent convergence validates that the simplified autoencoder effectively approximates the behavior of an ideal autoencoder without loss of generality. Of course, more sophisticated autoencoder implementations can fill the same role in future implementations of the RAAHN.

The simplified autoencoder component consists of a single layer of bidirectional weights that are trained to match the output of the backwards activation with the input to the forward activation. On each activation of the network, first the inputs I feed into the the regular forward activation of the hidden layer H (the input layer is fully connected to the hidden layer) in the following manner. For each hidden neuron j, forward activation A_j is calculated:

$$A_j = \sigma \left(\sum_{i \in I} (A_i \cdot w_{i,j}) + b_j \right), \tag{2}$$

where σ is a sigmoid function, A_i is the value of input neuron $i \in I$, $w_{i,j}$ is the weight of the connection between neurons i and j, and b_j is the bias on hidden neuron j. Next, the

forward activation values for hidden layer H are used to calculate the *backwards activation* to input layer I. For each input neuron i, backward activation B_i is calculated:

$$B_i = \sigma \left(\sum_{j \in H} (A_j \cdot w_{i,j}) + b_i \right), \tag{3}$$

where σ is the same sigmoid activation function as in equation 2, A_j is the forward activation on hidden neuron $j \in H$, and b_i is the bias on input neuron i.

After backwards activation is calculated, for each input neuron i, an error E_i is calculated:

$$E_i = A_i - B_i. (4)$$

Finally, as a simple heuristic for reducing reconstruction error (modeled after the perceptron learning rule), each weight is adjusted according to

$$\Delta w_{i,j} = \alpha E_i A_j,\tag{5}$$

where α is the learning rate (which is set to a small value to prevent convergence before an adequate number of input samples have been seen). This autoencoder was validated on data from the experiment to ensure that it converges with very low error (less than 5% from the optimal reconstruction). It is important again to note that any autoencoder could substitute for this simple model, whose particular mechanics are not essential to the overall RAAHN.

The experiment in this paper applies the proposed RAAHN system to a simulated robot control task. On each simulated time tick, the agent perceives values on its sensors and experiences a network activation. Thus, each time tick constitutes one training sample for the purpose of training the autoencoder connections. In this paper, a batch-style training system is implemented in which training samples are added to a history buffer of size n and autoencoder training is applied several times on the entire history buffer every $\frac{n}{2}$ ticks. Batch-style training is selected because many popular autoencoder training methods such as L-BFGS require batch training, although in preliminary experiments the system was found to perform well with both large and small values of n.

Hebbian Component

In the RAAHN system, connections between learned features and the outputs are trained with a modulated Hebbian learning rule, which is similar to the simple Hebbian rule (equation 1) with an added term to allow for the influence of reward and penalty signals. In this way, connections are only strengthened when a reward signal is received and when a penalty signal is received, the rule switches to anti-Hebbian (which serves to weaken connections). The modulated Hebbian rule is

$$\Delta w_i = m\eta x_i y,\tag{6}$$

where m is the modulation associated with the training sample. The Hebbian rule without modulation is like assuming that all training samples are positive; modulation allows training samples to be marked with varying degrees of positive or negative signal, which is a more flexible learning regime. The details of the modulation scheme have a significant impact on the effectiveness of learning. In the maze-navigating experiment in this paper, a simple modulation scheme is selected in which modulation is positive when the robot turns away from a wall, negative when the robot turns towards a wall, and neutral (m=0, corresponding to no learning) when there is no change. Specifically, the modulation component in this paper is calculated as the difference between the front-pointing rangefinder sensor activation on the previous tick and on the current tick, normalized to the range -1 to 1.

Modulated Hebbian learning following equation 6 is applied to every connection of the Hebbian component of the RAAHN system on each tick. The system in essence learns correlations between the developing feature set discovered by the autoencoder component and an output pattern required for effective control. The Hebbian rule is useful as a learning mechanism to connect autoencoder features to outputs because it is invariant to starting conditions. Thus, if the pattern of features in the learned feature set changes for some reason (e.g. the nature of the task environment shifts significantly), the Hebbian component can simply learn new correlations for the new feature set, enabling calibration in real-time as the feature set itself is refined.

Experiment

To demonstrate the effectiveness of the proposed RAAHN system, a maze-navigating control task is introduced in which a robot agent must make as many laps around the track as possible in the allotted time. The challenges of the task are two-fold. The first and more trivial challenge is for the controller to avoid crashing into walls, which would prevent it from completing any laps at all (robots that crash into walls almost always remain stuck on the wall, preventing any further movement). The second challenge arises because there are multiple round-trip paths around the track (figure 1). In particular, the track consists of an optimal path with two attached "detours" - longer routes that lead the robot off the optimal path before re-joining it. There is one detour attached to the inner wall of the track as well as one detour attached to the outer wall of the track. Both detours take significantly longer to traverse than the optimal path; thus taking either detour (or both detours) reduces the amount of laps the robot can make in the allotted time.

Robots in this task have access to two different types of sensors (figure 2). First, robots are equipped with a set of 11 wall-sensing rangefinders, each 200 units in length (slightly longer than the narrowest portions of the track) and equally spaced across the frontal 180 degrees of the robot (with one rangefinder pointing directly towards the front). Notice also

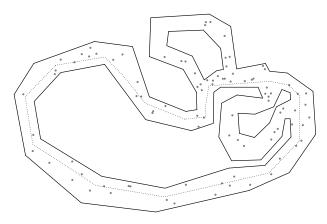
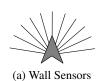


Figure 1: **Multiple path environment.** Robots navigate this cyclical track that consists of an optimal path (in terms of the shortest lap time) with two attached suboptimal detours. The training phase autopilot is denoted by a dotted line. Crumbs (gray dots) are scattered non-uniformly around the track to enable the identification of unique locations.

in figure 1 that there are "crumbs" scattered nonuniformly across the track. The random distribution of these crumbs means the robot can in principle identify unique locations. For this purpose, robots are equipped with 33 crumb-density sensors that sense the density of crumbs within a rangelimited pie slice. The crumb-density sensors are divided into three sets of 11 (near, mid, and far), each set equally spaced across the frontal 180 degrees of the robot. Near-type crumb density sensors sense crumbs between 0 and 132 units in distance, mid-type between 133 and 268 units, and far-type between 269 and 400 units. If one crumb is present within a crumb density sensor's area, then the sensor experiences 0.33 activation; it experiences 0.67 activation for two crumbs and 1.0 activation for three or more crumbs (it is rare for more than three crumbs to be present in a sensor's area). Robots have a single effector output, corresponding to the ability to turn right or left. Otherwise, robots are always moving forward at a constant velocity (5 units per tick).

In the experiment, robots first experience a *training phase*. During this phase an autopilot drives the robot around the track for 30,000 ticks. The robot is shown a path that never deviates onto suboptimal detours. However, the driving within the chosen path is not perfect. The autopilot, whose path (shown in figure 1) is deterministic, does not crash but it also does not always drive in the precise middle of the road; that way it is exposed to the penalty for moving too close to walls. During this training phase the autopilot overrides the robot's outputs, forcing it to move along the autopilot's route, while both the autoencoder and Hebbian learning are turned on. After the training phase, autopilot is turned off, learning is stopped, and agents are released to follow their learned controller for 10,000 ticks (the evaluation phase).

It is important to note that while this experiment could



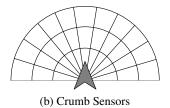


Figure 2: **Agent sensor types.** Robots have a set of 11 rangefinder sensors (a) for detecting the presence of walls (up to a maximum distance of 200 units). Robots also have an array of 33 non-overlapping range-limited pie slice sensors (b) to sense crumbs in the environment. Each sensor can detect up to three crumbs with increasing levels of activation, at which point sensor activation is capped. These crumb sensors form a semicircular grid up to a distance of 400 units across the frontal 180 degrees of the robot.

have been performed with a supervised learning framework, RAAHN is not restricted to supervised learning. Because RAAHN organizes its feature set and learns a control policy in real-time as it accumulates information about its environment, it can in principle perform when there is no autopilot training phase and robots are simply released into the world under their own control from the first tick. However, this type of application of RAAHN would require a more advanced modulation scheme that also rewards making laps and perhaps would require confronting the distal reward problem. In the interest of introducing the core learning mechanism without other potentially confusing variables, such a study is reserved for future investigation. It is also critical to note that the experiment even as devised is not supervised learning because the autoencoder is accumulating features in real-time with no error feedback whatsoever, just as would happen if the agent were allowed to control its own movements while learning. The autopilot simply ensures that the experience of the robot is consistent in this initial study so we can understand what is typically learned by a RAAHN when experience is consistent (though of course from different initial random starting weights).

Preliminary experiments revealed that a robot controller consisting of only rangefinder sensors connected directly to the output with Hebbian connections (i.e. without an autoencoder) was able to navigate the track with a trivial wallfollowing behavior that keeps close to and parallel to a wall on either the right or the left side while moving forward around the track. Because there is a detour attached to both the inner wall and the outer wall, robots performing such a trivial wall-following behavior will inevitably take one of the two detours each lap. The optimal behavior, which involves *avoiding* both detours while moving around the track at maximum speed and avoiding crashing into walls, therefore requires extra information and neural structure than

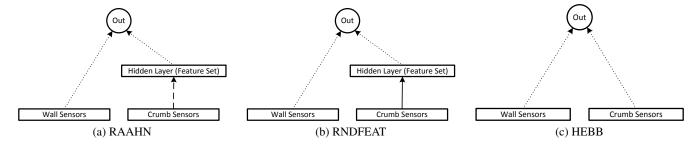


Figure 3: **Variant network structures.** Three variant networks are compared in the main experiment. Individual neurons within sensor array layers and hidden layers have been omitted for clarity. Layers shown as connected are fully connected. Dotted lines represent connections trained with modulated Hebbian plasticity. Dashed lines represent connections trained as an autoencoder. Solid lines represent static (non-trained) connections.

Hebbian learning from the raw rangefinder information. For the purposes of the main experiment, the crumb sensors and a layer of features extracted from the crumb sensors serve as this extra information. With the crumb sensors, robots in principle have the ability to distinguish between different parts of the track that have different "density signatures" (e.g. the opening for the outer detour causes a very different activation pattern on the crumb sensors than the opening for the inner detour). This distinguishing information makes it possible to enact different policies at different parts of the track (e.g. switching to following walls on the left side rather than the right side after passing the outer detour going around the track clockwise), which is essential for avoiding both detours and proceeding around the track along the optimal path. Thus an optimal agent must somehow encode these higher-level features.

The main experiment consists of a comparison between three very different learning regimes: RAAHN, RNDFEAT (random features), and HEBB. These methods differ only in the way that they process the extra information from the crumb sensors; all three methods include direct Hebbian connections from the wall-sensing rangefinders to the output. **RAAHN** (figure 3a) includes an autoencoder-trained feature set of 7 neurons drawn from the 33 crumb sensors. This feature set then feeds into the output via Hebbian connections. **RNDFEAT** (figure 3b) has the same structure as RAAHN, except autoencoder training is turned off. This configuration means that RNDFEAT has a set of 7 static random features. Results for RNDFEAT with 30 and 100 random features are also included (as RNDFEAT30 and RNDFEAT100, respectively), which resemble the idea behind extreme learning machines (Huang and Wang, 2011). Finally, **HEBB** (figure 3c) consists of all inputs directly connected to the output via Hebbian connections. In all variants, connection weights are randomly initialized with a uniform distribution of small magnitude (with average magnitude equal to 5% of the maximum weight of 3.0); increasing the magnitude of initial weights in preliminary experiments did not significantly impact the results.

Experimental Parameters

Batch autoencoder training occurs every 800 ticks on a history buffer of training samples spanning the past 1,600 ticks, which is roughly equivalent to the amount of time required to make a full lap during the training phase (recall that training encompasses 30,000 total ticks). The learning rate α for autoencoder training is 0.001. Each time batch training occurs, the buffer of training samples is spun through 10 times, with backwards activations recalculated after each pass. The result is that the autoencoder mostly converges by the end of a single training pass (autoencoder error is reduced to less than 5% of the optimal reconstruction). The Hebbian learning rate η for all variants is 0.2. These parameter settings were found to be robust to moderate variation through preliminary experimentation.

Results

In the results reported in this section, performance is based on the kinds of paths followed over the 10,000 tick evaluation period, averaged across 1,000 trials. Learned behaviors were found to be consistent, that is, the behavior observed during the first lap is very similar to the behavior on all other laps, especially with respect to which detours are taken (if any). Therefore, it is possible to place behaviors that navigate the maze into two key categories: First are robots that follow the optimal path; these best-performing robots never take a detour and thereby stay on the best path. Second are wallavoiding robots who deviate from the optimal path by taking at least one detour but still effectively avoid crashing and thereby make several clean laps around the track. Learned behaviors that do not fall into either of these two categories do not complete the course because they crash into walls. Experimental results are reported in figure 4 as the percentage of robots trained with each scheme that fall into either

The main result is that the RAAHN learns the optimal path four times more often than the closest variant (RNDFEAT30). HEBB never learns the optimal path. However, the results

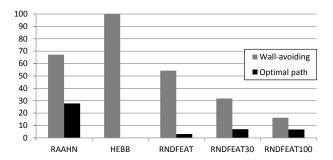


Figure 4: **Percentage of wall-avoiding and optimal path followers.** Results for each variant are percentages calculated over 1 000 evaluations.

confirm that pure Hebbian learning (HEBB) is invariant to starting conditions – it performs exactly the same in each of 1,000 trials (each with different initial weights). HEBB robots always follow the inner detour. Both RAAHN and RNDFEAT variants experience some failed trials, though RAAHN experiences the least. These failures come in many forms: robots that get stuck on sharp corners when exiting a detour are a common observed failure. Some robots fail immediately after leaving autopilot by spinning in tight circles – this type of failure (immediate failure) is not consistent across all methods. Interestingly, immediate failure occurs in 1.7% of RAAHN runs, 15.3% of RNDFEAT runs, 33.4% of RNDFEAT30 runs, and 58.6% of RNDFEAT100 runs.

Discussion and Future Work

The HEBB variant, which attempts to incorporate the crumb sensor inputs directly as a static feature set, never achieves optimal behavior. Thus, raw crumb sensor information appears to be insufficient for Hebbian learning to discover optimal behaviors on this task, suggesting the necessity of higher-level features. While HEBB's performance could in principle be manipulated by manually changing the set of static features (e.g. preprocessing the inputs), doing so is like the human playing the role of the RAAHN. This paper focuses instead on the automated generation of features with the intent of building a learning system that is generally applicable to future domains that may be too complicated for manual preprocessing of features or where adaptation of the feature set is required (e.g. large, open-ended artificial life worlds).

One naive way to automatically generate higher level features is through a hidden layer with random incoming weights, such as in RNDFEAT. While such random features bring optimal behaviors in this task into the realm of possibility, they do not constitute a strong method for achieving such behaviors – even the best RNDFEAT-type variant (RND-FEAT30) achieves the optimal behavior only 7% of the time. Adding more random features (as in RNDFEAT100) only causes performance to deteriorate. Indeed, the rate of immediate failures increases sharply as more random features are added to the feature set (compared to 7 features in RND-

FEAT), which suggests that additional random features are destructive as they wash out the useful signal from the wall sensors (e.g. leading to many immediate failures).

RAAHN does not suffer from the pathology of immediate failures because training the feature set as an autoencoder encourages the discovery of a more balanced feature set. RAAHN also discovers optimal behaviors four times as often (28%) as the best RNDFEAT-type variant (RND-FEAT30), which demonstrates the autoencoder's ability to learn useful information about the environment. Furthermore, RAAHN's feature set is dynamic, enabling it in principle to adapt to changing environmental conditions, while RND-FEAT's feature set cannot. Future RAAHNs might even add new layers over time, vielding a kind of real-time deep learning for control. A deeper analysis of neural activity within the autoencoder layer as well as experiments with larger quantities of autoencoded features will indicate the extent of the RAAHN's potential to be expanded in real-time in this way. Furthermore, as the state of the art in Hebbian learning advances, such as by addressing distal rewards (Soltoggio and Steil, 2013), the RAAHN benefits from the advancing capabilities as well.

Finally, while 28% optimality still may appear to leave room for improvement, it is important to note that it is actually impressive for the simple training regimen in this experiment. In particular, during training, robots were never actually shown the detours that they are expected to avoid, so there can be little to no representation of the inside of detours within the autoencoder. Furthermore, robots were not explicitly rewarded for taking the optimal path or penalized for taking detours. Rather, the reward scheme only rewards steering away from walls. Thus optimal paths were acquired entirely implicitly through observing the path taken by the autopilot and encoding its key features, suggesting the power of the RAAHN to derive behaviors from such features based on sparse and indirect feedback. In the future, when RAAHN-controlled networks are released to explore completely on their own while rewards are experienced, they have the potential to acquire a significantly wider breadth of abilities.

Conclusion

The experiment in this paper showed that a Hebbian layer can learn during ongoing behavior in real time from an autoencoder placed below it under controlled conditions. The implication of this initial step is that the RAAHN is a synergistic union that opens up many opportunities for new investigations. For example, what is possible to achieve when the RAAHN is allowed to acquire new features while exploring on its own without an autopilot? Can increasingly complex skills be acquired if the depth of the autoencoder is allowed to expand during learning? Can ANNs be evolved to incorporate both autoencoders and Hebbian plasticity? These are among the intriguing possibilities created by the RAAHN.

Acknowledgments

This work was partially supported through a grant from the US Army Research Office (Award No. W911NF-11-1-0489). This paper does not necessarily reflect the position or policy of the government, and no official endorsement should be inferred. This work was also partially supported by the European Communitys Seventh Framework Programme FP7/2007-2013, Challenge 2 Cognitive Systems, Interaction, Robotics under grant agreement No. 248311 - AMARSi.

References

- Baxter, J. (1992). The evolution of learning algorithms for artificial neural networks. In Green, D. and Bossomaier, T., editors, *Complex Systems*, pages 313–326. IOS Press, Amsterdam.
- Bednar, J. A. and Miikkulainen, R. (2003). Self-organization of spatiotemporal receptive fields and laterally connected direction and orientation maps. In De Schutter, E., editor, Computational Neuroscience: Trends in Research, 2003, pages 473–480.
- Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions* on pattern analysis and machine intelligence, pages 1798– 1928.
- Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. In Advances in Neural Information Processing Systems 19 (NIPS), Cambridge, MA. MIT Press.
- Bienenstock, L. E., Cooper, L. N., and Munro, P. W. (1982). Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *The Journal of Neuroscience*, 2(1):32–48.
- Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4-5):291–294.
- Cireşan, D., Meier, U., Gambardella, L., and Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural computation*, 22(12):3207–3220.
- Cireşan, D., Meier, U., Masci, J., and Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338.
- Coleman, O. J. and Blair, A. D. (2012). Evolving plastic neural networks for online learning: review and future directions. In AI 2012: Advances in Artificial Intelligence, pages 326–337. Springer.
- Floreano, D. and Urzelai, J. (2000). Evolutionary robots with online self-organization and behavioral fitness. *Neural Networks*, 13:431–4434
- Hebb, D. O. (1949). The Organization of Behavior: A Neuropsychological Theory.
- Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.
- Hinton, G. E. and Zemel, R. S. (1994). Autoencoders, minimum description length, and helmholtz free energy. *Advances in neural information processing systems (NIPS 1994)*.
- Huang, G.-B. and Wang, D. H. (2011). Extreme learning machines: a survey. *International Journal of Machine Learning & Cybernetics*, 2(107-122).
- Le, Q., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G., Dean, J., and Ng, A. (2012). Building high-level features using

- large scale unsupervised learning. In *International Conference* in *Machine Learning (ICML-2012)*.
- Marc'Aurelio, R., Boureau, L., and LeCun, Y. (2007). Sparse feature learning for deep belief networks. In *Advances in Neural Information Processing Systems 20 (NIPS)*, pages 1185–1192, Cambridge, MA. MIT Press.
- Niv, Y., Joel, D., Meilijson, I., and Ruppin, E. (2002). Evolution of reinforcement learning in uncertain environments: A simple explanation for complex foraging behaviors. *Adaptive Behavior*, 10(1):5–24.
- Oja, E. (1982). A Simplified Neuron Model as a Principal Component Analyzer. *Journal of Mathematical Biology*, 15(3):267–273.
- Ranzato, M., Poultney, C., Chopra, S., and LeCun, Y. (2006). Efficient learning of sparse representations with an energy-based model. In et al., J. P., editor, *Advances in Neural Information Processing Systems (NIPS 2006)*, volume 19. MIT Press.
- Risi, S., Hughes, C., and Stanley, K. (2011). Evolving plastic neural networks with novelty search. *Adaptive Behavior*, 18(6):470– 491.
- Risi, S. and Stanley, K. O. (2010). Indirectly encoding neural plasticity as a pattern of local rules. In *Proceedings of the 11th International Conference on Simulation of Adaptive Behavior (SAB2010)*, Berlin. Springer.
- Risi, S. and Stanley, K. O. (2012). A unified approach to evolving plasticity and neural geometry. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN-2012)*, Piscataway, NJ. IEEE.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel Distributed Processing*, pages 318–362.
- Soltoggio, A., Bullinaria, A. J., Mattiussi, C., Drr, P., and Floreano, D. (2008). Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. In Bullock, S., Noble, J., Watson, R., and Bedau, M., editors, *Proceedings of the Eleventh International Conference on Artificial Life (Alife XI)*, Cambridge, MA. MIT Press.
- Soltoggio, A., Dürr, P., Mattiussi, C., and Floreano, D. (2007). Evolving neuromodulatory topologies for reinforcement learning-like problems. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2471–2478.
- Soltoggio, A. and Jones, B. (2009). Novelty of behaviour as a basis for the neuro-evolution of operant reward learning. In Proceedings of the 11th Annual conference on Genetic and evolutionary computation, GECCO '09, pages 169–176, New York, NY, USA. ACM.
- Soltoggio, A., Lemme, A., Reinhart, F. R., and Steil, J. J. (2013). Rare neural correlations implement robotic conditioning with reward delays and disturbances. *Frontiers in Neurorobotics*, 7:6(Research Topic: Value and Reward Based Learning in Neurobots).
- Soltoggio, A. and Stanley, K. O. (2012). From modulated hebbian plasticity to simple behavior learning through noise and weight saturation. *Neural Networks*, 34:28–41.
- Soltoggio, A. and Steil, J. J. (2013). Solving the distal reward problem with rare correlations. *Neural computation*, 25(4):940– 978.
- Stanley, K. O., Bryant, B. D., and Miikkulainen, R. (2003). Evolving adaptive neural networks with and without adaptive synapses. In *Proceedings of the 2003 Congress on Evolutionary Computation*, Piscataway, NJ. IEEE.

JBotEvolver: A Versatile Simulation Platform for Evolutionary Robotics

Miguel Duarte^{1,2}, Fernando Silva^{1,3}, Tiago Rodrigues^{1,2}, Sancho Moura Oliveira^{1,2}, and Anders Lyhne Christensen^{1,2}

¹Instituto de Telecomunicações, Lisboa, Portugal

²Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal

³LabMAg, Faculdade de Ciências, Universidade de Lisboa, Portugal

miquel_duarte@iscte.pt

This paper introduces JBotEvolver, a versatile simulation platform for research and education in evolutionary robotics (ER). JBotEvolver is a Java-based open-source, cross-platform framework available at https://code.google.com/p/jbotevolver/ under the GNU GPL. JBotEvolver has been used in a number of previous ER studies of our research group, from offline evolution to online evolution and learning, and from single to multirobot systems (for examples see Duarte et al. (2014a,b); Silva et al. (2012a,b)), and in a number of undergraduate and graduate courses at ISCTE-IUL.

JBotEvolver's main features are its ease of installation and use, and its versatility in terms of customization and extension. A fundamental design philosophy behind JBotEvolver is to provide a basis for ER experiments without the need for detailed framework-specific knowledge. Following this philosophy, JBotEvolver enables the configuration of experiments programmatically or via a plaintext file that specifies which features will be included in the simulation. The corresponding classes are then seamlessly loaded in execution time via Java's Reflection API. In this way, JBotEvolver can also make use of external, user-defined classes that extend the base implementation. Additionally, JBotEvolver is selfcontained, but can also be used as an external library in other applications. One example is the automation system that allowed us to evolve hierarchical controllers by automatically combining controllers from independent evolutionary setups (Duarte et al., 2014a).

The user can extend the main components of JBotE-volver such as the environments in which the robots operate, the physical objects of the environment, the robot models, the evaluation functions, the evolutionary algorithms, and the type and structure of the controllers. For instance, while our studies have focused on the evolution of neural network-based controllers (Duarte et al., 2014b,a; Silva et al., 2012a,b), JBotEvolver does not preclude other approaches such as genetic programming or evolutionary fuzzy systems. JBotEvolver features a 2D differential-drive kinematics engine that has been used to simulate multirobot systems with up to thousands of robots (Duarte et al.,

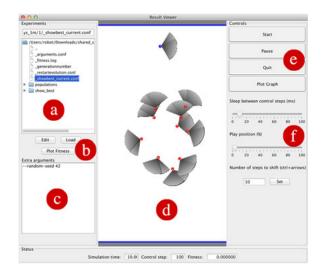


Figure 1: Result Viewer GUI: (a) file tree for navigating experimental results, (b) options to load or edit files, and to plot the fitness of controllers, (c) text area for overriding experimental arguments, (d) basic 2D visual renderer, (e) options to start/stop the experiments, quit the simulator, and plot neural network activity, and (f) sliders to change the speed of the simulation and to fast-forward/rewind.

2014b). The 2D engine can also be extended or replaced by user-defined engines with different dynamics. With respect to the experimental setup, JBotEvolver allows the user to control the degree of realism of the simulations by allowing, for instance, the use of sensors modelled based on samples from real robots (Duarte et al., 2014a) and the configuration of the robots' control cycle frequency. In terms of robots, JBotEvolver includes a model of the e-puck robot and allows for fully customizable 2D robot models.

An important characteristic of JBotEvolver is that controller evaluations are defined as tasks that can be executed sequentially or in parallel on a workstation. We have also included a connector for Conillon (Silva et al., 2011), a lightweight platform for distributed computing that enables a significantly speed-up of evolutionary processes through

Features Simulator	Simbad	Webots	ARGoS	Player	Enki	JBotEvolver
2D/3D	3D	2D+3D	2D+3D	2D+3D	2D	2D
Open-source	yes	no	yes	yes	yes	yes
Dependencies	Java 3D	multiple	multiple	multiple	multiple	none
Platforms	win/mac/linux	win/mac/linux	mac/linux	mac/linux	mac/linux	win/mac/linux
Language	Java	multiple*	C++	C++	C++	Java
Learning curve	low	intermediate	high	high	intermediate	low
Distributed evolution	no	no	no	no	no	yes
GUI	rich	rich	medium	medium	basic	rich

Table 1: Comparison of features between simulators. *Webots allows development in C, C++, Java, Python, Matlab, and URBI.

parallelization. We have used Conillon to distribute JBotE-volver tasks to over 400 cores. Conillon's dynamic request of Java classes allows tasks with different codebases to be submitted. Worker nodes can be added to the computing network in an ad-hoc manner either through: (i) a standalone application, (ii) a Java applet running in a browser, or (iii) as a screensaver on PCs. In addition, the Encog framework¹ implementation of the neuroevolutionary NEAT algorithm (Stanley and Miikkulainen, 2002) has been interfaced with JBotEvolver.

A number of alternative simulators for evolutionary robotics are available, including: (i) Simbad (Hugues and Bredeche, 2006), (ii) Webots (Michel, 2004), (iii) AR-GoS (Pinciroli et al., 2012), (iv) Enki², and (v) the Player Project (Gerkey et al., 2003), which includes the 2D simulator Stage and the 3D simulator Gazebo. In comparison with such platforms, as described in Table 1, the key advantages of JBotEvolver are its extensibility, ease of use, and versatility. JBotEvolver has a number of expert-oriented features, such as the mechanisms for the distributed execution of experiments, and non-expert-oriented features. From the GUI (Fig. 1), it is, for instance, possible to navigate between results of different evolutionary runs or setups, analyze fitness score plots, visualize the controllers' input and output values, stop and resume experiments, and modify experimental configurations on-the-fly.

To summarize, JBotEvolver is a versatile, easy to deploy and operate simulation platform intended for both expert and non-expert users. In our ongoing work, we continue to use JBotEvolver, and we are extending it to support different robot models and types of robots, including aquatic drones.

Acknowledgements This work was partially supported by Fundação para a Ciência e Tecnologia under the grants SFRH/BD/76438/2011, SFRH/BD/89573/2012, PEst-OE/EEI/LA0008/2013, PEst-OE/EEI/UI0434/2014, and EXPL/EEI-AUT/0329/2013.

References

Duarte, M., Oliveira, S. M., and Christensen, A. L. (2014a). Evolution of hybrid robotic controllers for complex tasks. *Journal of Intelligent and Robotic Systems*. In press.

Duarte, M., Oliveira, S. M., and Christensen, A. L. (2014b). Hybrid control for large swarms of aquatic drones. In *Proceedings of the Fourteenth International Conference on the Synthesis & Simulation of Living Systems (ALIFE)*. MIT Press, Cambridge, MA. In press.

Gerkey, B., Vaughan, R. T., and Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the Eleventh International Conference on Advanced Robotics (ICAR)*, pages 317–323. FCT/UC, Coimbra, Portugal.

Hugues, L. and Bredeche, N. (2006). Simbad: an autonomous robot simulation package for education and research. In Proceedings of the Ninth International Conference on the Simulation of Adaptive Behaviour (SAB), pages 831–842. Springer, Berlin, Germany.

Michel, O. (2004). Webots: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):39–42

Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., Ducatelle, F., Birattari, M., Gambardella, L., and Dorigo, M. (2012). ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. Swarm Intelligence, 6(4):271–295.

Silva, F., Urbano, P., and Christensen, A. L. (2012a). Adaptation of robot behaviour through online evolution and neuro-modulated learning. In *Proceedings of the Thirteenth Ibero-American Conference on Artificial Intelligence (IBERAMIA)*, pages 300–309. Springer, Berlin, Germany.

Silva, F., Urbano, P., Oliveira, S., and Christensen, A. L. (2012b). odNEAT: An algorithm for distributed online, onboard evolution of robot behaviours. In *Proceedings of the Thirteenth International Conference on the Simulation & Synthesis of Living Systems (ALIFE)*, pages 251–258. MIT Press, Cambridge, MA.

Silva, H., Oliveira, S. M., and Christensen, A. L. (2011). Conillon: A lightweight distributed computing platform for desktop grids. In *Proceedings of the Sixth Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6. IEEE Press, Piscataway, NJ.

Stanley, K. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.

¹Encog: http://www.heatonresearch.com/encog

²Enki: http://home.gna.org/enki/

Systematic Derivation of Behaviour Characterisations in Evolutionary Robotics

Jorge Gomes^{1,2} and Pedro Mariano² and Anders Lyhne Christensen^{1,3}

¹Instituto de Telecomunicações, Lisbon, Portugal

²LabMAg - Faculdade de Ciências da Universidade de Lisboa, Portugal

³Instituto Universitário de Lisboa (ISCTE-IUL), Lisbon, Portugal

jgomes@di.fc.ul.pt, plmariano@fc.ul.pt, anders.christensen@iscte.pt

Abstract

Evolutionary techniques driven by behavioural diversity, such as novelty search, have shown significant potential in evolutionary robotics. These techniques rely on priorly specified behaviour characterisations to estimate the similarity between individuals. Characterisations are typically defined in an ad hoc manner based on the experimenter's intuition and knowledge about the task. Alternatively, generic characterisations based on the sensor-effector values of the agents are used. In this paper, we propose a novel approach that allows for systematic derivation of behaviour characterisations for evolutionary robotics, based on a formal description of the agents and their environment. Systematically derived behaviour characterisations (SDBCs) go beyond generic characterisations in that they can contain task-specific features related to the internal state of the agents, environmental features, and relations between them. We evaluate SDBCs with novelty search in three simulated collective robotics tasks. Our results show that SDBCs yield a performance comparable to the task-specific characterisations, in terms of both solution quality and behaviour space exploration.

Introduction

Evolutionary robotics (ER) is focused on the application of evolutionary algorithms to the synthesis of robot controllers, and sometimes robot morphologies (Nolfi and Floreano, 2000). In ER, the evolutionary process is typically driven towards solutions for a task according to a manually crafted fitness function (Nelson et al., 2009). Evolutionary algorithms driven by a fitness function are, however, vulnerable to deception and convergence to local optima (Whitley, 1991). Common approaches to overcome these issues are based on diversity preservation. Traditionally, diversity is preserved at the genomic level (Goldberg and Richardson, 1987). However, in ER it has been observed that many different genotypes can represent similar behaviours, and that small changes in the genotype can lead to substantial changes in behaviour (Mouret and Doncieux, 2012). Therefore, recent works have proposed the use of behaviour similarity measures (BSM) in the evolutionary process to promote an effective diversity in the population.

A number of different strategies using BSM have been proposed. The most popular approach, novelty search, directly rewards individuals that display novel behaviours (as measured by the BSM), instead of using a fitness function (Lehman and Stanley, 2011; Mouret and Doncieux, 2012). BSM have also been used to speciate the population according to the behaviour of the individuals (Trujillo et al., 2011; Moriguchi and Honiden, 2010). Ollion and Doncieux (2011) defined a measure for the degree of behaviour space exploration, based on BSM, and used the measure to predict the relative performance of different evolutionary setups.

Devising effective behaviour measures is not straightforward (Savage, 2004), since the measure must be able to capture the details of the behaviour that are relevant in the given task, while at the same time should contain no or as few irrelevant behaviour features as possible. Behaviour measures are typically defined by the experimenter based on task-specific knowledge (for examples, see Mouret and Doncieux, 2012; Lehman and Stanley, 2011; Mouret and Doncieux, 2009; Gomes et al., 2013). Relying on the experimenter's intuition about the task may, however, introduce biases in the evolutionary process. These biases can compromise the evolutionary search, as the experimenter might not know beforehand what type of behavioural traits should be explored in order to solve the task.

Several researchers have proposed the use of *generic* BSM (Doncieux and Mouret, 2010; Gomez, 2009; Gomes and Christensen, 2013) for ER. Generic measures do not rely on the specific details of a task, and can be used throughout the domain. Their use is, however, associated with two key issues: (i) generic measures are only weakly related with the specific task, which can create an unnecessarily large behaviour space (Cuccu and Gomez, 2011), and (ii) they are mostly unintelligible, undermining analysis and comprehension of the behaviour space exploration.

We propose a new class of behaviour measures, *system-atically derived behaviour characterisations* (SDBCs), that combine the advantages of task-specific and generic measures. The proposed measures are directly derived from a formal description of the task state. This way, we reduce the dependency on the experimenter's knowledge about the task while, at the same time, obtain characterisations that are

directly related to the task. In this paper, we apply SDBCs to robotics tasks, but the approach could potentially be generalised to other types of agent-based systems, as the formal task description resembles the SMART agent framework (d'Inverno et al., 2004). We augment our method with the inclusion of a weighting scheme for behaviour features, based on the estimated relevance of the features.

The proposed measures are evaluated with novelty search in three simulated collective robotics tasks: resource sharing, gate escape, and predators-prey pursuit. We compare the proposed measures with behaviour characterisations defined based on task-specific knowledge.

Related Work

Task-specific distance measures

Most previous works on behavioural diversity rely on behaviour characterisations designed specifically for the given task. These characterisations are composed of behavioural traits that the experimenter considers relevant for describing agent behaviour in the context of the given task. The behavioural distance then corresponds to the Euclidean distance between the characterisation vectors. Table 1 lists several examples of characterisations that have been used in previous ER studies. Analysing these characterisations, the following commonalities can be identified:

- There is a strong focus on the spatial relationships between entities in the task environment (1, 2, 5, 6, 8, 9).
- Characterisations often comprise only a small number of different behavioural traits (1, 3–5, 7–9).
- Many characterisations focus on the final state of the environment (1, 3–5, 7–9), values averaged over an entire trial (7–9), or a single quantity sampled over time (2, 6).

Generic distance measures

Gomez (2009) proposed the use of generic measures for assessing behaviour similarity. In the proposed approach, action records for the agents are compared, using either the Hamming distance, relative entropy, or normalised compression distance (NCD). Doncieux and Mouret (2010) extended generic measures to evolutionary robotics, proposing the following BSMs:

Hamming distance: Distance between the sequence of all the binary sensor and effector values of the agent sampled through time.

DFT: A discrete Fourier transform is applied to the sensor-effector sequence, and the first coefficients are used to compute the distances between individuals.

State count: Each possible sensor-effector state corresponds to one entry in a vector. Each entry contains the number of times the corresponding state was visited.

Gomes and Christensen (2013) extended the *Hamming distance* and *state count* measures, making them applicable to multiagent systems and non-binary sensors and effectors.

Common to all generic measures is that they rely exclusively on the sensor-effector states of the agents. While this design makes the measures generic and widely applicable, as any robotics experiment involves robots with sensors and effectors, it also represents their main weakness. Since there is no direct relationship between a given task and the generic behaviour characterisation, the behaviour space is typically very large, which can compromise the performance of diversity-based techniques (Cuccu and Gomez, 2011; Mouret, 2011). Additionally, the characterisations are mostly unintelligible, since they are not intrinsically related to observable characteristics of the agent's behaviour.

Mouret and Doncieux (2012) compared task-specific and generic characterisations in a comprehensive empirical study with a number of single-robot tasks. Doncieux and Mouret (2013) showed how different similarity measures (generic or task-specific) can be combined, by either switching between them throughout evolution or by calculating the behaviour distance based on all similarity measures.

Novelty search

To evaluate the proposed SDBCs, we use novelty search to evolve controllers for three distinct multirobot tasks. In novelty search (Lehman and Stanley, 2011), individuals are scored according to a *novelty metric*, instead of how well they perform a given task.

The novelty metric quantifies how different an individual is from other, previously evaluated individuals with respect to behaviour. The metric relies on the mean behaviour distance (as given by the BSM) of that individual to the k nearest neighbours. Potential neighbours include the other individuals of the current population and a sample of individuals from previous generations stored in an archive. Candidates from sparse regions of the behaviour space therefore tend to receive higher novelty scores, generating a constant evolutionary pressure towards behavioural innovation.

As novelty search is guided by behavioural innovation alone, its performance can be greatly affected by the size and shape of the behaviour space. In particular, behaviour spaces that are vast or contain dimensions not related with the task can cause novelty search to perform poorly (Cuccu and Gomez, 2011; Mouret, 2011; Gomes et al., 2012). To address this issue, we augment novelty search and include a fitness objective, as suggested in (Mouret, 2011; Mouret and Doncieux, 2012).

Systematically Derived Behaviour Characterisations – SDBC

In the analysis of the task-specific behaviour characterisations used in previous works, it can be observed that the characterisations are generally composed of relatively simple behavioural features (see Table 1). Based on these regularities, we formalise a workflow for the definition of behaviour characterisations. We show how such workflow can

Table 1: Behaviour characterisations used in previous works in evolutionary robotics. The right-most column lists the lengths
of the characterisation vectors, with S denoting that the length is proportional to the simulation length.

Study	Task	Characterisation	Length	
(Lehman and Stanley, 2011a;		1. Final location of the robot		
Mouret and Doncieux, 2012; Maze navigation Mouret, 2011)				
(Lehman and Stanley, 2011a)	Biped locomotion	2. Robot trajectory	S	
(Trujillo et al., 2011)	Homing navigation			
(Lehman and Stanley, 2011b)	Virtual creature evolution	3. Height, mass, number of joints	3	
(Mourat and Danaious, 2012)	Sequential light seeking	4. Whether each of the seven light switches was activated		
(Mouret and Doncieux, 2012)	Ball collecting robot	5. Final position of the balls	8	
(Compared 2012)	Aggregation	6. Sampled mean distance to the centre of mass (and/or number of clusters)		
(Gomes et al., 2013)	Recharging station sharing	7. Number of survivors, mean energy, (mean speed, mean distance to station)		
(Gomes et al., 2014)	Predators vs prey pursuit	8. Capture state, time to capture, final distance to prey, mean dispersion		
(Gomes et al., 2014)	Keepaway soccer	9. Number and mean length of passes, game length, mean dispersion		

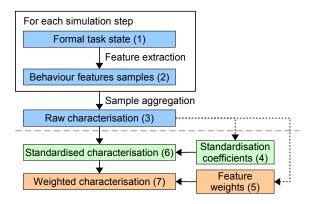


Figure 1: Workflow of the SDBC approach for obtaining the behaviour characterisation of each individual. The steps below the dashed line are performed after the entire population is evaluated. See the text for details.

be used to systematically derive behaviour characterisations with minimal dependence on the experimenter's understanding of the task.

The proposed workflow is summarised in Figure 1. The interface between the specific task and the derivation of the behaviour characterisations is accomplished through a formal description of the current state of the task (1) that should be specified by the experimenter. At each simulation step, a set of behaviour features is extracted from the formal task state (2). After the simulation has ended, the behaviour feature samples are combined to assemble a fixed-length raw characterisation vector (3). Based on all the raw characterisation vectors obtained in the current population, a set of standardisation coefficients is calculated (4), and optionally, a set of feature weights (5). The raw characterisations are then transformed with the standardisation coefficients (6) and feature weights (7). The behaviour distance between individuals is then given by the Euclidean distance between the respective transformed characterisation vectors. Below, we present the task description formalism and details on each step in the workflow.

Task state formalism

The task state formalism separates the specific task details from the method used to devise behaviour characterisations, thus reducing the dependence on the experimenter.

The proposed formalism makes no distinction between agents and environmental features, both are treated equally as *entities* of the task. Each entity $e \in E_{\kappa}$ is associated with a κ -tuple of state attributes (ϑ) , and a tuple x with the constant properties of the entity. The tuple ϑ is composed of the properties of the entity that can change during task execution. To encompass multiagent tasks, we introduce the concept of *entity group*. An entity group $g \in G$ can be formed by an arbitrary number of entities that share the same type and number (κ) of attributes. The cardinality (η) of a group can change during the task, and a single agent can constitute a group. The experimenter should provide the minimum (η_{min}) and maximum (η_{max}) size of each group.

A task state is composed of (i) a list of entity groups, and (ii) a distance function f_D . The function f_D should measure or estimate the physical distance between any two entities. A task state T is thus defined by:

$$T = \langle g_1, \cdots, g_N, f_D \rangle , g_i \in G$$

$$G = \{ \langle e_1, \cdots, e_\eta \rangle \mid e_i \in E_\kappa \land \eta \in [\eta_{min}, \eta_{max}] \}$$

$$E_\kappa = \{ \langle \vartheta, x \rangle \mid \vartheta \in \mathbb{R}^\kappa \}$$

$$f_D : E_\kappa \times E_{\kappa'} \to \mathbb{R}$$

$$(1)$$

Extraction of behaviour features

At each time step, we automatically extract a set of behaviour features from the formal task state. These features correspond to a high-level description of the agents' behaviour and environment's state. They measure spatial relations between agents of the same entity group and between different groups, and the mean state of each group. Previous work (Gomes et al., 2013) has shown that averaging behaviour features over a group's agents can lead to effective and scalable characterisations for multiagent systems. We defined the following behaviour features:

• Size of each entity group, relative to the respective limits:

$$S_g = \frac{|g| - \eta_{min_g}}{\eta_{max_g} - \eta_{min_g}}, g \in G \land \eta_{max_g} > \eta_{min_g}$$
 (2)

• For each entity group, mean state of the entities:

$$\vartheta_g = \left\langle \sum_{e \in g} \frac{\vartheta_e[1]}{|g|}, \cdots, \sum_{e \in g} \frac{\vartheta_e[\kappa]}{|g|} \right\rangle, g \in G \quad (3)$$

 For each entity group, mean distance of each entity to the other entities of the same group (a measure of dispersion):

$$D_g = \sum_{e_i \in g} \sum_{e_j \in g, j \neq i} \frac{f_D(e_i, e_j)}{(|g| - 1)^2} , g \in G \land |g| > 1 \quad (4)$$

• Mean pairwise distance between each two entity groups:

$$R = \langle d(g_1, g_2), \cdots, d(g_1, g_N), \cdots, d(g_{N-1}, g_N) \rangle$$

$$d(g_a, g_b) = \sum_{e_i \in g_a} \sum_{e_j \in g_b} \frac{f_D(e_i, e_j)}{|g_a||g_b|}, a \neq b$$
(5)

Aggregation of feature samples

After a simulation has ended, the samples obtained for each behaviour feature at each time step are aggregated to assemble a fixed-length characterisation vector. Inspired by the task-specific characterisations used in previous work (see Table 1), a concatenation of the mean and final values of each behaviour feature is performed. The duration of the simulation is additionally included as a feature in the characterisation.

Feature standardisation

The behaviour characterisations can be composed of features that have different nature and thus have very distinct ranges. To overcome this disparity, the characterisations of the current population are standardised with the same set of coefficients at every generation. If a novelty archive exists, it also needs to be updated with those coefficients. Each behaviour feature k of a vector b is standardised according to:

$$b_k' = (b_k - \mu_k)/\sigma_k \quad , \tag{6}$$

where μ_k and σ_k are respectively the mean and standard deviation of feature k.

Feature weighting

Previous work has shown that behaviour features weakly related with the task might be harmful (Gomes et al., 2013). We therefore propose and evaluate a method for weighting the behaviour features according to their estimated relevance to the task. Relevance is estimated based on *mutual information feature selection (MIFS)* (Battiti, 1994) — a machine learning feature selection algorithm. Mutual information is a quantitative measurement of how much one random variable tells us about another random variable. In the proposed

approach, the relevance of each behaviour feature is estimated according to the mutual information between the feature values and the fitness scores of the individuals. Since the fitness score typically reflects the degree of fulfilment of a given task, this measure estimates the relevance of each behaviour feature with respect to the solution of the task.

Each behaviour feature is assigned a weight equal to the mutual information estimate plus a minimum weight δ . Therefore, the higher the mutual information, the higher the impact of that behaviour feature in the distance measure. The added minimum feature weight guarantees that none of the behaviour features is completely ignored. Each feature k of a characterisation b is weighted according to:

$$b_k' = b_k \cdot [\delta + I(F;k)] \quad , \tag{7}$$

where I(F;k) is an estimate of the mutual information between the fitness scores (F) and the behaviour feature k, and δ is the minimum feature weight. We set $\delta=0.25$ in all experiments, which is relatively small when compared to the typical mutual information scores of the more relevant features. Values of 0, 0.1 and 0.5 were also tested, but they yielded no improvements.

Behaviour feature weights have to be periodically updated throughout evolution. For scalability purposes, we only use the current population to calculate the weights. The computational complexity associated with weights calculation could be further reduced by performing updates less frequently, every n generations, for instance.

Experimental Setup

We use three different tasks to evaluate the general applicability of our approach. The performance of SDBCs is compared with task-specific characterisations and fitness-based evolution. For the resource sharing task and predator-prey pursuit, we use task-specific characterisations that have been fine-tuned in previous works. In all tasks, the robots are homogeneous, i.e., the same neural network is copied to each robot of the group for evaluation. Each controller is evaluated in 10 simulations with randomised initial conditions.

Gate escape task

In this task, a group of robots must escape through a narrow gate that closes shortly after the first robot has passed. To solve the task successfully, the robots must first find the gate, and then wait for each other before starting to pass through the gate. The task is detailed in Figure 2 (top-left) and Table 2. The fitness function F_q is given by:

$$F_g = \frac{g + t/\tau}{1 + N} \ , \tag{8}$$

where g is the number of robots that escaped through the gate, t is the simulation length, τ is the maximum simulation length, and N is the total number of robots. The task-specific behaviour characterisation is a vector of length four

comprising the following behavioural features (normalised to [0,1]): (i) number of escaped robots; (ii) time when the gate opened; (iii) mean distance to the gate; and (iv) mean dispersion of the robots.

The task state T_g is composed of three entity groups: (i) the group of the robots that have still not passed through the gate; (ii) the gate; and (iii) the walls. Each robot has five attributes: its x and y position, turning speed, linear speed, and whether it is currently passing through the gate or not. The gate has one state variable, denoting whether it is closing or not. The surrounding walls are stateless.

Resource sharing task

In the resource sharing task (Gomes et al., 2013; Gomes and Christensen, 2013), a group of robots must coordinate in order to allow each member periodical access to a single battery charging station. The energy consumption varies with the motor speed, and the charging station can only hold one robot at a time. The task is detailed in Figure 2 (top-right) and Table 2. The fitness function F_s is given by:

$$F_s = \frac{s + \overline{e}/e_{max}}{1 + N} \quad , \tag{9}$$

where s is the number of surviving robots, \overline{e} is the mean energy of the robots throughout the simulation, and N is the number of robots. The task-specific characterisation was proposed in (Gomes et al., 2013): a vector of length four, composed of: (i) the number of surviving robots; (ii) the mean energy of the robots; (iii) the mean movement speed; and (iv) the mean distance of the robots to the charging station. Each of these elements is normalised to [0,1].

The task state T_s is composed by two entity groups: (i) the group of alive robots; and (ii) the charging station. Each robot has six attributes: its x and y position, turning speed, linear speed, current energy level, and whether it is currently charging. The charging station has a single state variable indicating whether it is currently charging a robot or not.

Predator-prey pursuit

In the predator-prey task, three predators have the objective of capturing a single prey, i.e., one predator should physically touch the prey. The predators can sense one another. The simulation ends if the prey escapes the chase zone. Only the predators' controller is evolved, and the behaviour of the prey is preprogrammed: the prey moves away from nearby predators at full speed, and stops moving when it does not sense any predators. The task is detailed in Figure 2 (bottom) and Table 2. The fitness function F_p is given by:

$$F_p = \begin{cases} 2 - t/\tau & \text{if prey captured} \\ max(d_i - d_f, 0)/size & \text{otherwise} \end{cases} , (10)$$

where t is the simulation length, τ is the maximum length, d_i is the mean initial distance from the predators to the prey, d_f

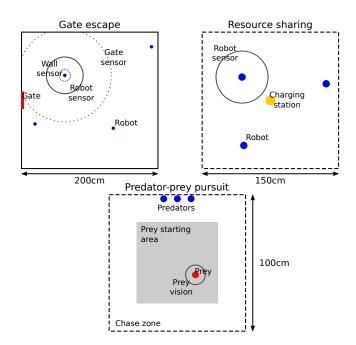


Figure 2: Initial simulation conditions. In the predators-prey task, the predator's initial position is fixed. In the other two tasks, the robots are placed randomly in the environment.

Table 2: Parameters of the tasks and the evolutionary setup.

Gate escape task		Predator-prey pursuit		
Gate location sensor	50 cm	Prey location sensors	∞	
Wall detection sensors	4 x 5 cm	Predator location sensors	∞	
Robot detection sensors	4 x 25 cm	Predators' max. speed	5 cm/s	
Nearby robot counter	25 cm	Prey's escape distance	5 cm	
Robot's max. speed	5 cm/s	Prey's escape speed	7.5 cm/s	
Gate close time	15 s	Max. simulation length	60 s	
Max. simulation length	200 s			
Resource sharing	task	Genetic algorithm	m	
Robot detection sensors	4 x 25 cm	Crossover	none	

Resource sharing task				
4 x 25 cm				
∞				
-				
-				
5 cm/s				
500 u				
[5,10] u				
100 u/s				
100 s				

Tournament size	5
Population size	200
Novelty search	
Novelty nearest k	15
Add archive probability	3%
Maximum archive size	1000

Gene mutation rate

is the mean final distance, and size is the arena's diagonal. The task-specific characterisation (Gomes et al., 2014) is a vector of length four, with all elements normalised to [0,1]: (i) whether the prey was captured or not; (ii) the simulation length; (iii) the mean final distance of the predators to the prey; and (iv) the mean distance of the predators to their centre of mass (dispersion) throughout the simulation.

The task state T_p is composed of three entity groups: (i) the group of the predators; (ii) the prey; and (iii) the chase zone boundaries. Both the predators and the prey have the same attributes: x and y location, turning speed, and linear speed. The environment is stateless.

Evolutionary setup

We used a canonical generic algorithm, where the neural network weights are directly encoded in the chromosomes. The parameters of the genetic algorithm and novelty search are listed in Table 2. All evolutionary treatments use the same parameters. The individuals are scored with a multi-objectivisation of novelty and fitness scores: at each generation, the individuals are sorted according to their Pareto front and crowding distance (Deb et al., 2002), considering the fitness and novelty objectives.

Results and Analysis

Quality of the solutions

We compare fitness-driven evolution (*Fit*) to novelty search with the three behaviour characterisation approaches: task-specific characterisations (*NS-TS*), SDBC with feature weighting (*NS-SD*+), and SDBC without weighting (*NS-SD*). Figure 3 shows the highest fitness scores achieved with each approach, in each of the three tasks.

The distribution of the highest fitness scores achieved by Fit displays a high variance, since all tasks have some degree of deception and Fit often converges to local optima. NS is unaffected by deception, and all variants achieve significantly higher fitness scores than Fit (Mann-Whitney U test, p-value < 0.05). The scores achieved by NS-TS are on average superior to NS-SD+ in the gate escape (p-value = 0.013) and resource sharing tasks (p-value = 0.025). The difference is, however, relatively small, and visual inspection of the solutions revealed no clear difference between the best controllers evolved by NS-TS and NS-SD+.

Across the three tasks, the use of feature weighting never harms the performance of SDBCs, and NS-SD+ is on average superior to NS-SD. Significant differences between NS-SD and NS-SD+ are, however, only present in the resource sharing task (p-value < 0.01). What makes NS-SD+ superior in this particular task is not evident, and additional experiments are needed to clarify the potential advantage provided by the weighting scheme. In the following sections, we analyse the weighting scheme and what impact it has on behaviour exploration.

Relevance of behaviour features

Following Equations 2–5, and the task descriptions presented in the Experimental Setup, a total of 10 behaviour features were extracted for the gate-escape task, 10 features for the resource sharing task, and 13 features for the predator-prey pursuit. After the feature sample aggregation process, this resulted in behaviour characterisations of lengths 21, 21, and 27, respectively.

Since feature extraction is systematic, it is important to determine if the behaviour features can adequately characterise the behaviour of the individuals in the context of the given task. Table 3 lists the mutual information scores of the

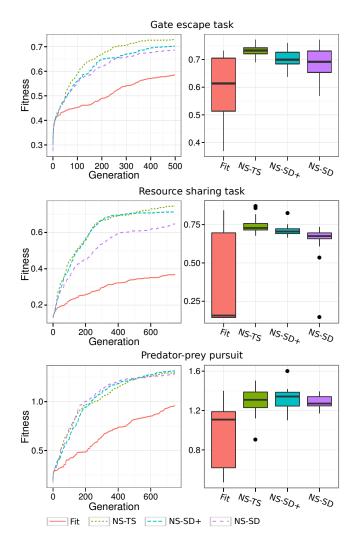


Figure 3: Left: highest fitness scores found at each generation, averaged over 20 evolutionary runs for each method and task. Right: boxplots of the highest fitness scores found in each evolutionary run. The whiskers are the extreme values within 1.5 IQR and the dots are outliers.

most relevant features for each task. It can be observed that some of the behaviour features match the features defined in the task-specific characterisations (highlighted in the table with a tick). The remaining features correspond to behavioural traits that are intuitively related with the fulfilment of the task. The relevance of each feature is also consistent across the multiple evolutionary runs, as indicated by the relatively small standard deviations of the means.

Our results suggest that the mutual information between feature values and fitness scores is a good indicator of the relative relevance of each behaviour feature to task fulfilment. Furthermore, the analysis shows that the systematically derived characterisations include highly relevant behaviour features for the given tasks. This explains why the performance of novelty search with systematically derived

Table 3: Mutual information (MI) between behaviour features and fitness scores, averaged over each evolutionary run (*Mean MI*). The standard deviation is shown in the *SD MI* column. Only the features with highest MI scores are shown. (*F*) marks features measured at the final state, while (*M*) marks features averaged over simulation time. The rightmost column indicates if a similar feature is present in the corresponding task-specific characterisation.

Feature	Mean MI	SD MI	In TS char.		
Gate escape task					
Gate is closing (F)	2.37	0.05			
Agent group size (F)	1.85	0.03	\checkmark		
Simulation length	1.46	0.04	\checkmark		
Agent group size (M)	1.43	0.03			
Gate is closing (M)	1.39	0.04			
Agent is passing gate (M)	0.81	0.04			
Resource sharing task					
Agent energy level (M)	1.58	0.14	√		
Agent group size (F)	1.56	0.17	\checkmark		
Agent energy level (F)	1.12	0.09			
Simulation length	0.96	0.06			
Station is occupied (M)	0.84	0.08			
Agent is charging (M)	0.81	0.07			
Predator-prey prey pursuit					
Predators-prey distance (F)	1.47	0.05	√		
Predators-prey distance (M)	0.98	0.01			
Predators dispersion (F)	0.66	0.08	\checkmark		
Prey speed (F)	0.57	0.03			
Prey-bounds distance (F)	0.48	0.02			

characterisations is very similar to novelty search with characterisations specifically designed for each task.

Behaviour space exploration

We analysed the exploration of the behaviour space to determine the underlying differences between the novelty search variants. The behaviour space was built with the SDBC features. We used a Kohonen map to reduce the dimensionality of the space for visualisation purposes. The resulting plots for the resource sharing task can be seen in Figure 4. The results for the other two tasks are similar.

Fitness-based evolution typically explores only a narrow region of the behaviour space, which translates to premature convergence and in turn leads to an inferior performance. All novelty search variants explore the behaviour space more uniformly. The behaviour exploration in *NS-SD*+ follows a pattern similar to *NS-TS*, but with a slightly inferior focus on the high-fitness behaviour region (highlighted in the maps). Regarding the influence of the weighting scheme, it is clear that *NS-SD*+ spends more effort in the high-fitness region, when compared to *NS-SD*. The behaviour exploration of *NS-SD* is the most uniform since there are no bias towards

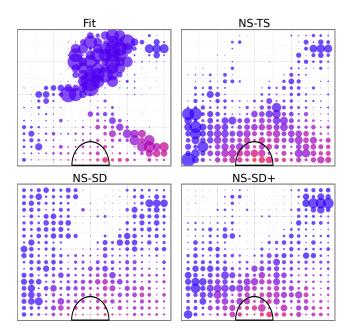


Figure 4: Average behaviour space exploration in the resource sharing task. The bigger the circles, the higher the exploration in that specific behaviour region. The behaviour region associated with the highest fitness scores is highlighted with a semicircle (bottom-middle).

specific behaviour dimensions. The absence of bias causes *NS-SD* to explore a behaviour region to which none of the other methods devoted significant effort (top-left corner in the maps). These results suggest that although there was no significant difference between *NS-SD*+ and *NS-SD* in terms of solutions' quality (see Figure 3), the weighting scheme can actually focus behaviour exploration in the most relevant regions.

Conclusion

We proposed an approach for systematically deriving behaviour characterisations (SDBCs) for evolutionary robotics. The proposed approach relies on a formal description of the task state. Behaviour features are systematically extracted based on the state of the agents, their environment, and the spatial relationships between the physical entities of the task. We also proposed a feature weighting scheme that estimates the relevance of the extracted features, based on the mutual information between feature values and fitness scores. We demonstrated the proposed approaches with novelty search, using three different simulated collective robotics tasks.

Our results showed that SDBCs are on par with task-specific characterisations, with the advantage of relying less on the experimenter's task-specific knowledge. The quality of the evolved solutions is similar, and the behaviour space exploration with task-specific characterisations was similar to SDBCs with the weighting scheme. Analysing the most

relevant behaviour features present in SDBCs, we could observe that they either match features of the task-specific characterisation or correspond to behavioural traits that are highly related to solving the task. While the calculated relevance scores are in accordance with our understanding of the tasks, using these scores to weight the characterisations did not translate in a significant advantage in terms of the fitness scores achieved. Our results did, however, show that feature weighting helps focus the exploration on more relevant behaviour regions and more effort is spent in regions associated with high fitness scores.

In light of our results, we consider that the proposed systematic approach represents a promising way of defining behaviour characterisations. SDBCs reduce the dependency on the experimenter's intuition about the task, thus introducing fewer biases in behaviour exploration. Nevertheless, the systematically derived characterisations contain behaviour features that are highly related to the specific task, and can accurately capture the behaviour of the individuals. The approach is flexible and extensible, and can potentially accommodate the specific details of many tasks, possibly even outside the domain of embodied agents.

Possible extensions of the proposed approach include: (i) extraction of additional features from the task state, e.g., by measuring inter-group relations between attributes of the same type; (ii) improved strategies for combining the behaviour feature samples obtained during task execution, e.g., through function approximation or time discretisation; and (iii) more elaborate weighting schemes, that could for example eliminate redundancy between behaviour features.

AcknowledgementsThisresearchissupportedbyFundaçãoparaaCiênciaeTecnologia(FCT)grantsPEst-OE/EEI/LA0008/2013,PEst-OE/EEI/UI0434/2014,SFRH/BD/89095/2012 and EXPL/EEI-AUT/0329/2013.

References

- Battiti, R. (1994). Using mutual information for selecting features in supervised neural net learning. *Transactions on Neural Networks*, 5(4):537–550.
- Cuccu, G. and Gomez, F. (2011). When novelty is not enough. In *Applications of Evolutionary Computation*, pages 234–243. Springer.
- Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Transactions on Evolutionary Computation*, 6(2):182–197.
- d'Inverno, M., Luck, M., and Luck, M. M. (2004). *Understanding* agent systems. Springer.
- Doncieux, S. and Mouret, J.-B. (2010). Behavioral diversity measures for evolutionary robotics. In *Congress on Evolutionary Computation*, pages 1–8. IEEE Press.
- Doncieux, S. and Mouret, J.-B. (2013). Behavioral diversity with multiple behavioral distances. In *Congress on Evolutionary Computation*, pages 1427–1434. IEEE Press.

- Goldberg, D. E. and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In *Conference* on *Genetic Algorithms*, pages 41–49. Lawrence Erlbaum.
- Gomes, J. and Christensen, A. L. (2013). Generic behaviour similarity measures for evolutionary swarm robotics. In *Genetic and Evolutionary Computation Conference*, pages 199–206. ACM Press.
- Gomes, J., Mariano, P., and Christensen, A. L. (2014). Avoiding convergence in cooperative coevolution with novelty search. In *International Conference on Autonomous Agents and Multi-agent Systems*, pages 1149–1156. IFAAMAS.
- Gomes, J., Urbano, P., and Christensen, A. (2012). Progressive minimal criteria novelty search. In *Ibero-American Confer*ence on Artificial Intelligence, pages 281–290. Springer.
- Gomes, J., Urbano, P., and Christensen, A. L. (2013). Evolution of swarm robotics systems with novelty search. *Swarm Intelligence*, pages 115–144.
- Gomez, F. J. (2009). Sustaining diversity using behavioral information distance. In *Genetic and Evolutionary Computation Conference*, pages 113–120. ACM Press.
- Lehman, J. and Stanley, K. O. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223.
- Moriguchi, H. and Honiden, S. (2010). Sustaining behavioral diversity in neat. In *Genetic and Evolutionary Computation Conference*, pages 611–618. ACM Press.
- Mouret, J.-B. (2011). Novelty-based multiobjectivization. In *New Horizons in Evolutionary Robotics*, pages 139–154. Springer.
- Mouret, J.-B. and Doncieux, S. (2009). Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity. In *Congress on Evolutionary Computation*, pages 1161–1168. IEEE Press.
- Mouret, J.-B. and Doncieux, S. (2012). Encouraging behavioral diversity in evolutionary robotics: an empirical study. *Evolutionary Computation*, 20(1):91–133.
- Nelson, A. L., Barlow, G. J., and Doitsidis, L. (2009). Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57(4):345–370.
- Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics: The Biology, Intelligence, and Technology*. MIT Press.
- Ollion, C. and Doncieux, S. (2011). Why and how to measure exploration in behavioral space. In *Genetic and Evolutionary Computation Conference*, pages 267–274. ACM Press.
- Savage, T. (2004). Measurement and the explanation of adaptive and novel behaviors in real and artificial creatures. *Cognitive Systems Research*, 5(1):3–39.
- Trujillo, L., Olague, G., Lutton, E., De Vega, F. F., Dozal, L., and Clemente, E. (2011). Speciation in behavioral space for evolutionary robotics. *Journal of Intelligent & Robotic Systems*, 64(3-4):323–351.
- Whitley, L. D. (1991). Fundamental principles of deception in genetic search. In *Foundations of Genetic Algorithms*, pages 221–241. Morgan Kaufmann.

Soft Robotics and Morphologies

Evolved Electrophysiological Soft Robots

Nicholas Cheney¹, Jeff Clune², and Hod Lipson¹

¹ Creative Machines Lab, Cornell University. Ithaca, NY

² Evolving Artificial Intelligence Lab, University of Wyoming. Laramie, WY nac93@cornell.edu, jeffclune@uwyo.edu, hod.lipson@cornell.edu

Abstract

The embodied cognition paradigm emphasizes that both bodies and brains combine to produce complex behaviors, in contrast to the traditional view that the only seat of intelligence is the brain. Despite recent excitement about embodied cognition, brains and bodies remain thought of, and implemented as, two separate entities that merely interface with one another to carry out their respective roles. Previous research co-evolving bodies and brains has simulated the physics of bodies that collect sensory information and pass that information on to disembodied neural networks, which then processes that information and return motor commands. Biological animals, in contrast, produce behavior through physically embedded control structures and a complex and continuous interplay between neural and mechanical forces. In addition to the electrical pulses flowing through the physical wiring of the nervous system, the heart elegantly combines control with actuation, as the physical properties of the tissue itself (or defects therein) determine the actuation of the organ. Inspired by these phenomena from cardiac electrophysiology (the study of the electrical properties of heart tissue), we introduce electrophysiological robots, whose behavior is dictated by electrical signals flowing though the tissue cells of soft robots. Here we describe these robots and how they are evolved. Videos and images of these robots reveal lifelike behaviors despite the added challenge of having physically embedded control structures. We also provide an initial experimental investigation into the impact of different implementation decisions, such as alternatives for sensing, actuation, and locations of central pattern generators. Overall, this paper provides a first step towards removing the chasm between bodies and brains to encourage further research into physically realistic embodied cognition.

Introduction and Background

The fields of evolutionary robotics and artificial life have seen a great deal of emphasis on embodied cognition in recent years [Cheney et al. (2013); Bongard (2013); Rieffel et al. (2013); Auerbach and Bongard (2012); Hiller and Lipson (2012a); Lehman and Stanley (2011); Auerbach and Bongard (2010a,b); Pfeifer et al. (2007); Hornby et al. (2001); Lipson and Pollack (2000)]. There is even a paradigm called *embodied cognition*, which argues that the specifics of the embodiment (such as the morphology) are

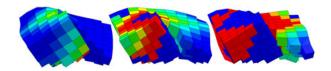


Figure 1: Current flowing through an evolved creature. The legend for voltage within each cell (colors) is given in Fig. 3.

vital parts of the resulting behavior of the system: It argues that the co-evolutionary connection between body and brain is more deeply intertwined than the body simply acting as a minimal physical interface between the brain and the environment [Pfeifer and Bongard (2006)].

Recent work in evolutionary robotics has shown that complex behaviors can arise when co-evolving bodies and brains. At one end of the spectrum, Auerbach and Bongard (2010b) demonstrated the evolution of physical structures that had no joints or actuators, and evolved to cover the largest distance in a controlled fall due to gravity. While that work exemplifies the evolution of behavior emerging from morphology alone, it does not co-evolve any actuation or control. Auerbach and Bongard (2010a) then evolved the placement of CPG controlled rotational joints between cellular spheres, thus co-evolving morphology and control.

Cheney et al. (2013) evolved locomoting soft robots made of multiple different materials: two passive voxels of differing rigidity and two actuated voxel types that expanded cyclically via out-of-phase central pattern generators (CPGs). While this work added a variety of soft materials and a new type of actuation, the pairing of muscle types directly to a CPG again reflected a focus on evolving morphology rather than sophisticated neural control.

Many examples in the literature include the co-evolution of a robot morphology with an artificial neural network controller [Sims (1994); Lipson and Pollack (2000); Hornby et al. (2001); Lehman and Stanley (2011)]. These studies (and many more like them) involve what might be called "ghost" networks: artificial neural networks that provide control to the body, yet do not have any physical embodi-

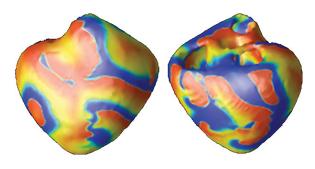


Figure 2: An example of complex electrical wave propagation in cardiac modeling [Fenton et al. (2005)].

ment in the system they control. The state of input nodes to these networks is often set by sensors in the robot and output nodes typically signify behavioral outcomes in the actuators, but the computation is done supernaturally, disjoint from the body itself.

In the age of 3D printing, it is a realistic goal for robots to physically walk out of a printer. It is thus worthwhile to consider designing robots that can be physically realized: i.e., those whose controllers are accounted for by being physically woven into the design of the robot.

While the brains of animals are often a separate module within their bodies, animals also have central and peripheral nervous systems extending throughout their bodies. An extreme example of this is the octopus, which has as much as 90% of its neurons existing outside of its central nervous system [Zullo et al. (2009)]. The distributed and physical layout of the nervous system over space may contribute significantly to neural processing, as the delays and branching in axons (the basis for nerves) are suggested to serve computational functions [Segev and Schneidman (1999)].

Despite the prevalence of embodied, distributed circuitry in nearly all of animal life, the idea of an embodied nervous system has been absent from the field of evolutionary robotics. The sub-field called Evolvable Hardware evolves physical circuits for computer chips [Floreano and Mattiussi (2008)], but such work has not been applied to evolving the circuitry of artificial life organisms. We are unaware of work with virtual creatures that have physically embodied control systems (e.g. where neural circuitry physically runs throughout the body of the creature). We present the first such work in this paper.

We propose a very basic model of electrical signal propagation throughout the body of an evolved creature. This embodied controller is based on electrophysiology (specifically at large scales, such as cardiac electrophysiology, Fig. 2). Electrophysiology is the study of the electrical properties of biological cells and tissues [Hoffman et al. (1960)]. In this model, electrical pulses from a single centralized sinusoidal pacemaker (analogous to the sinoatrial node – the pacemaker in the heart [Brown (1982)]) are propagated through the

electrically conductive tissue of the creature. The location and patterning of this conductive tissue is described by an evolved Compositional Pattern Producing Network (CPPN) genome. Evolution controls the shape of the body and the electrical pathways within it, which both combine to determine the robot's behavior.

The model involves conductive tissue cells that collect voltage from neighboring cells, causing an action potential (spike) if the collected voltage exceeds the cell's firing threshold (Fig. 3). Once this threshold is crossed, the cell depolarizes, causing a voltage spike that excites neighboring cells. This voltage spike is followed by a refractory period, during which the cell is temporarily unable to be re-excited.

This model allows for the propagation of information through the body of the creature in the form of electrical signals. The structure of this flow is produced entirely by the topology of the creature and the state of each cell's direct neighbors. In this sense, the model can be seen as a form of distributed information processing. One could draw similarities between this model and a 3D-grid of neurons, where each neuron receives inputs from, and has outputs to, its immediate neighbors. In this analogy, we are evolving where neurons should exist in the grid, what type of material the neuron is housed in, as well as the material type, if any, of grid locations that do not contain neurons.

The placement of material, which is under evolutionary control, directly determines the resultant behavior of the organism. Cells that actuate will contract and expand as they depolarize (much like the contraction of cardiac muscles), leading to the locomotion behavior of the creature. In order to control the signal flow throughout the creature, insulator cells are allowed, which are unable to accept and pass on the signal. Evolution can also choose not to fill a voxel with material. The morphology of the simulated robot and tissue type at each cell is determined by a CPPN genome.

This model examines the evolution of embodied cognition at a more detailed level of implementation than is typical in the literature – with embodied control circuitry resulting directly from the morphology of the individual creature. While this study only covers the classic problem of locomotion, it is a step towards truly physically embodied robots.

Methods

CPPN-NEAT

The evolutionary algorithm employed in this study is CPPN-NEAT. This algorithm has been previously described in detail (Stanley, 2007, 2006; Auerbach and Bongard, 2010b; Cheney et al., 2013), so it is only briefly described here.

A Compositional Pattern Producing Network (CPPN) [Stanley (2007)] is variation of an Artificial Neural Network (ANN) [McCulloch and Pitts (1943); Floreano and Mattiussi (2008)] where each node can have one of many mathematical functions as an activation function (e.g. sine, cosine, Gaussian, sigmoid, linear, square, or positive square root)

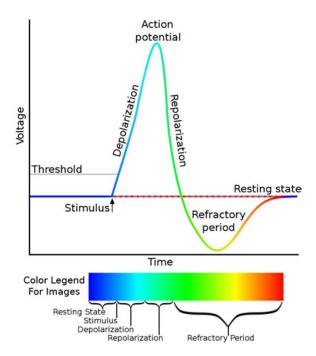


Figure 3: (top) A depiction of an action potential. Notice how the voltage is below the threshold until a stimulus event (such as a pacemaker or neighboring cell spike) pushes the voltage to the threshold value. Once this threshold is met, a voltage spike occurs via a process called depolarization. The cell excites its immediate neighbors during this process. Following the action potential, the cell enters a fixed length refractory period, during which it is physically unable to produce a new action potential. Finally, the cell returns to its resting state, able to start the process again when a new stimulus arrives. (bottom) These phases of the action potential cycle are mapped to the color code used to visualize the soft robots in Figs. 1, 4, and 5. Image licensed via Creative Commons.

instead of being limited to a sigmoid activation function. In CPPN-NEAT, a design space is discretized into individual locations (in this case a 3D space is discretized into a $10 \times 10 \times 10$ grid of voxels, for 1000 total voxels). The CPPN is queried once per voxel to determine the phenotypic state at that location (in this case, whether a voxel is present and, if so, the material type). The inputs to the CPPN network for each location are different: specifically, they include one input node for each dimension of the space (here, reporting the x, y, and z values of that location), as well an input that reports the distance (d) from the center to the location. The network also features output nodes for each material property. There are three in this study: one node specifies if a voxel exists at the queried location, the second decides if the material at that location is conductive, and the third decides whether or not the material is an actuated muscle (the latter two only matter if the voxel is present).

Conductive VoxCad

Fitness is evaluated in the VoxCad soft body simulator [Hiller and Lipson (2014)]. Its dynamics have previously allowed the evolution of complex and lifelike behaviors in soft robots, as it can simulate muscle contractions [Cheney et al. (2013)]. Further details about VoxCad can be found in Hiller and Lipson (2012b).

This work adds electrophysiology to VoxCad by adding a simple action-potential model, acting on the scale of a single voxel (analogous to a cell). Each voxel has an immediate membrane potential level (the difference between the electric potential inside and outside the cell), as well as a threshold membrane potential level. In an action-potential model (Fig. 3), a cell's resting potential is below that of the threshold potential. When the membrane potential reaches its threshold value, the cell depolarizes, causing a spike in the cell's membrane potential and voltage.

Following the depolarization, the cell hyperpolarizes, dropping the voltage and membrane potential below their original values, as the cell enters a refractory period. During this refractory period, the cell is unable to be depolarized again. In biological cells, the refractory period also consists of a relative refractory period when the cell is able to be depolarized, but only by unusually high voltage levels. For simplicity, we ignore this aspect in our model, and consider only the absolute refractory period, during which depolarization is disallowed. This refractory period means that the current is unable to flow backwards towards recently depolarize cells, causing the unidirectional propagation of action potentials in a wave across the cells.

A cell's action potential (starting with the beginning of the depolarization phase in Fig. 3) triggers a sinusoidal expansion/contraction of that cell with a maximum amplitude of 39% linear expansion per voxel side.

A given cell may transmit current to any other cell that it is physically touching. In 3D, this rule means that up to 26 neighboring voxels (the "Moore neighborhood") can be activated by a single voxel. The threshold potential of each cell is set such that it will be excited if, and only if, at least one of its neighboring cells undergoes an action potential that causes that cell's voltage to spike. The time it takes a cell to excite its neighbor is half of its depolarization period. This delay in excitation means that the electric signal does not instantly activate all contiguously connected cells, but rather spreads outwards in a wave-like pattern of muscle actuation.

Cells may be of any of the following types: empty, conductive muscles, insulating muscles, conductive passive tissue, insulating passive tissue, or a pacemaker cell. Near the center point of the discretized $10\times10\times10$ design space, a lone pacemaker is placed (cell number 555 out of 1000). Analogous to the sinoatrial node in cardiac electrophysiology, this pacemaker node serves as the source of electric stimulation for the entire system. Insulating cells are similar to the cells explained above, except that they are unable to

accept current from neighboring cells and thus never reach their threshold potential or produce an action potential.

In this model, the refractory period lasts 5 times as long as the depolarization period. This means that at least 5 voxels must separate the leading edges of two serial action potential waves. Since the pacemaker is placed in the center of the $10 \times 10 \times 10$ space, approximately one wave of action potentials would exist at any given time in a setup with a uniform cube of entirely conductive material – where a wave of action potentials would propagate uninterrupted, with a new one starting around the time the first reaches the outer edge of the space. We chose this setup to encourage the evolution of static gaits, which can be more robust and transferable to reality than dynamic gaits [Belter et al. (2008)].

The length of the expansion/contraction period of each node is set equal to the refractory period, such that each cell is guaranteed to be fully returned to its original size before its next actuation cycle begins.

Task and Fitness Evaluation

Following Cheney et al. (2013), we evolve these electrophysiological robots for locomotion over flat ground. This simple task and environment make fitness evaluation easy. Despite its simplicity, the task is a classic problem in the field, and has been repeatedly shown to produce an array of complex morphologies and interesting behaviors [Cheney et al. (2013); Clune et al. (2009, 2011); Auerbach and Bongard (2014); Lehman and Stanley (2011)].

Each creature is simulated for 20 times the length of an expansion/contraction cycle. Its displacement between the starting coordinates and the creature's final center of mass (in the xy plane) is recorded. In an effort to discourage designs that might excite as many cells as possible, and to encourage designs with sparse spindles of connectivity (similar to the peripheral nervous system), the distance traveled is multiplied by $1 - \frac{(\# \ of \ conductive \ cells)}{1000}$. Thus the fitness function incentivizes minimizing the amount of conductive tissue and maximizing the distance traveled. While a multi-objective technique may be ideal in finding the optimal tradeoff between these goals, we follow previous CPPN-NEAT research in using this single, multi-part fitness function [Cheney et al. (2013); Auerbach and Bongard (2009)].

Experimental Parameters

Unless otherwise noted, each treatment described below consists of 48 independent runs (with identical initial conditions across treatments). Each run consists of a population size of 30 individuals evolved for 1000 generations. Unless otherwise noted, all other parameters are consistent with Cheney et al. (2013).

Statistical Reporting

Because the data are not normally distributed, all plots show median fitness (thick, center lines) bracketed by two thin lines that represent 95% bootstrapped confidence intervals of the median [Sokal and Rohlf (1995)]. For the same reason, all *p*-values are generated with the non-parametric Mann-Whitney-Wilcoxon Rank Sum test, which does not assume normality. Reported *p*-values compare the distance traveled by the top organism for each of the 48 runs at the final (1000th) generation. Plots report distance traveled, not adjusted fitness (which penalized for the number of conducting voxels as explained previously).

Results

Since this is the first study of evolved electrophysiological robots, there are many unanswered questions regarding the design and implementation of such a system. Many arbitrary design choices were made during the initial implementation. Here, we examine the impact of some of these choices.

As with many explorations in evolved virtual organisms, one of the main goals is complex, natural-appearing behavior. However, there are no satisfactory metrics for the "naturalness" or complexity of evolved behaviors. For this reason, we must rely on our qualitative, subjective assessments. A video of the evolved behaviors can be seen on the "Cornell Creative Machines Lab" Youtube channel, or found directly at this link: http://goo.gl/CvJp4. We believe the behaviors are interesting, complex, and lifelike – at least as much as in Cheney et al. (2013) – despite the added challenges of evolving physically embedded control.

We observed that physically instantiated control circuitry can produce both predictable and chaotic behaviors. Fig. 4 shows a simple wave of action potentials propagating outwards from the center of the creature, with little interruption. Fig. 5 reveals the evolution of unpredictable physical dynamics that still produce functional behavior. Notice the multiple "inputs" to a potential self-sustaining circular pathway. Fig. 1 demonstrates a circular actuation pattern of intermediate complexity, due more so to changes in the robot's shape than to material differences within it. We now turn to more quantitative analyses.

Pacemaker Placement

The placement for the pacemaker was an arbitrary decision made during the design of this new system. In an effort to mimic the midline location of the central nervous system in biology, the pacemaker was placed in the middle of the design space from which the creature was built. Thus action potential waves could propagate out equally in all directions and were not biased in any particular direction of travel. In order to test the effect of this arbitrary choice, a treatment was also performed where the pacemaker was located at the center voxel of the roof of the $10 \times 10 \times 10$ design space – voxel number 955 (where indices increase from the bottom, left hand, nearest corner), as well as a treatment that placed the pacemaker in the top right corner – voxel 999.

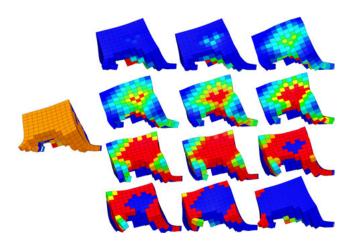


Figure 4: An action potential wave propagating across a mostly homogeneous surface. (left, single robot): The robot has a large patch of continuous conductive muscle on its back. In this pre-simulation state, cell colors signify the following: orange cells are conductive, blue cells are non-conductive; dark colors (blue or orange) signify muscle cells, while lighter colors (blue or orange) signify nonactuatable cell tissue; the red cell at the bottom is the robot's pacemaker cell. (right, 3 × 4 grid of robots): A progression over time (left to right, top to bottom) shows the wavelike propagation of the action potential phases (color meanings are described in Fig. 3). Note how the action potential emerges from the center, stimulated by the wave propagating out through the conductive tissue from the pacemaker below it. Following the light blue depolarization, the yellow and red phases show the longer lag of the refractory period, following in exactly the same pattern made by the leading edge of the action potential wave. As the wave fully passes, the cells return to their dark resting state and are thus able to spike again with a new action potential when the next wave comes.

As shown in Fig 6, the placement of this pacemaker significantly affects performance. While a central location (baseline treatment) shows significant advantages compared to the top-center and top-corner pacemaker locations ($p=4.91\times10^{-11}$ and 7.16×10^{-16} , respectively), a statistically significant difference is also demonstrated between the two less-different treatments: the top-center location outperformed the top-corner location ($p=3.43\times10^{-4}$). These results show that the pacemaker location can have a clear effect on the evolved behaviors. Future work shall place the exact location under evolutionary optimization.

Speed of Pacemaker

Another implementation decision was the low-frequency pacemaker to allow for static gaits. The increased stability and robustness of static gaits is appealing, and this may allow better transferability to physical robots (Belter et al.,

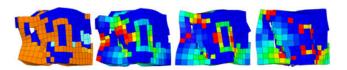


Figure 5: A more complex electrophysiological robot. (*left robot*): Contrary to Fig. 4, this creature shows complex patterning of the orange conductive tissue within the insulating blue tissue. (*right three robots*): As they unfold over time (left to right), the action potential waves in this robot produce a highly fractured, counter-intuitive actuation pattern that involves electrical signals flowing through long, sparse connective corridors and around corners (an explanation of the colors is provided in Figs. 3 and 4). The result is an interesting and unexpected behavioral pattern wherein the creature mashes and spins the left side of its body, which is separated from the larger, right side of its body by a large, oddly shaped internal cavity. Despite this bizarre behavior, it effectively locomotes. This behavior and others can be viewed on Youtube at: http://goo.gl/CvJp41.

2008). However, animals often employ dynamic gaits when there is an incentive for speed (as there is here). The tradeoff between these two is not known in this system. To examine this tradeoff, we compared three different treatments. First, the baseline treatment includes a pacemaker with the relatively slow pace of 40 beats per second (BPS). Since the baseline evaluation period is half a second, this results in 20 electrical pulses from the pacemaker per trial. A second treatment explores the increased potential for dynamic gaits at the maximum pacemaker speed of 80 BPS (the limit is due to the fixed length of the refractory period). In this faster treatment, each individual cell contracts at the same rate as before, but the pacemaker is now exciting cells as soon as their refractory period ends, instead of waiting (the length of an additional actuation cycle) before sending another pulse into the system. This system uses twice the amount of energy, producing 40 action potential waves in the same half second. In a third treatment, the faster paced (80 BPS) pacemaker is evaluated for half its normal time length, resulting in 20 beats per evaluation. This treatment allows a fair comparison of pacemakers in terms of distance traveled per "beat", rather than per unit time.

Unsurprisingly, the faster pacemaker evaluated for the full half second outperforms both the slower pacemaker evaluated for the same time period and the faster pacemaker evaluated for the shorter evaluation time ($p < 10^{-16}$ for both, Fig. 7). Interestingly, the frequency of the pacemaker has no significant effect on the distance traveled (p = 0.51 at generation 1000), suggesting that any disparity between the faster and slower gaits was not realized in simulation (with the number of beats held constant). Testing this result in the transfer to physical robots is a subject for future work.

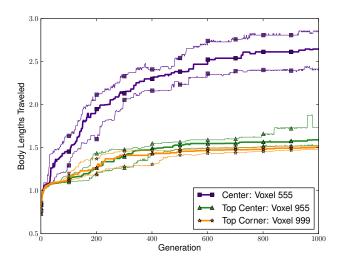


Figure 6: The effect of the placement of the central pattern generator (CPG) on the evolved speed. In one treatment, the CPG is placed at the top corner of the $10\times 10\times 10$ design space (voxel 999). This treatment performs slightly, but significantly ($p=3.43\times 10^{-4}$), better than another treatment that places the CPG at the center of the top plane of the bounding box (voxel 955). Outperforming both of these ($p<4.91\times 10^{-11}$) is the baseline treatment in which the CPG is always placed as close to the center of the bounding box as possible.

Touch Sensors

Another implementation decision was the use of pacemakers as the primary drivers of the system. While pacemakers, also known as central pattern generators, are biologically motivated [Ijspeert et al. (2007)], we could instead ask evolution to generate its own cadence. To provide an alternative to the pacemaker, we tested a treatment with touch sensors in lieu of a steady internal signal.

The touch sensors, like the pacemaker, are capable of producing an electrical signal. However, they do so in response to contact with the ground, rather than in a regular rhythm. In this treatment, all conductive cells have this touch-sensing ability and produce an action potential when in contact with the ground if not in the refractory period. Thus waves of action potentials propagate outwards from the touch sensors only when they are both in contact with the ground and fully recovered from their prior depolarization.

Thus, the upper bound on the number of action potentials that the touch sensors could produce is that of an 80 BPS pacemaker (the 80 BPS pacemaker fires again as soon as exiting the refractory period, where the touch sensors do so only if also touching the ground at that time – the slower 40 BPS pacemaker waits the length of one cycle before firing again). To reach this upper bound, touch sensors would have to be touching the ground exactly at the time when they com-

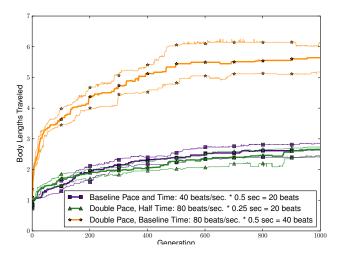


Figure 7: The effect of faster pacemakers (CPGs). It is not surprising that a faster CPG (80 beats per second) travels farther when evaluated for longer, or when compared to a slower CPG ($p < 10^{-16}$). However, when the the comparison is made according to distance per beat (half time/full speed against half speed/full time – both producing a total of 20 beats) there is no difference in their performance at Generation 1000 (p-value = 0.51), suggesting that CPG speed does not greatly affect evolved locomotion speed.

pleted their refractory period, and thus it is likely that this ceiling would not be reached in all cases. For a comparison, Fig. 8 shows the median distance traveled over evolutionary time plotted against that of the slower pacemaker (40 BPS) and the faster pacemaker (80 BPS) described above, and evaluated for the baseline half-second evaluation time. It is not surprising that the slower pacemaker falls behind the pack here, as it is handicapped by a throttle on its only source of action potentials compared to the faster pacemaker and the touch sensors ($p < 10^{-16}$). The tighter race is between the touch sensor and the faster pacemaker. In the early stages (< 150 generations), the robots with touch sensors significantly outperform robots with a pacemaker. However, in the later stages of evolutionary optimization, the touch sensor treatment shows modest gains compared to the continued innovation of the pacemaker treatment, with the pacemaker treatment significantly outperforming it at the end of the run $(p = 1.27 \times 10^{-7})$. The relatively low level of improvement in the touch sensor treatment in the later stages of evolution may suggest the premature convergence on local optima. The multiple points of origin for action potential waves, and thus wave collisions, may have also had an effect. An additional issue that could have hindered performance in this treatment is the upward propagation of signals from touch sensors on the ground, versus outwards expanding waves from the center of the organism.

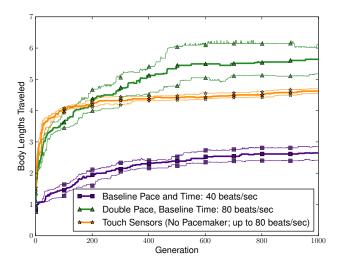


Figure 8: The performance of touch sensors vs. central pattern generators. The touch sensor treatment produces an expected number of beats with the upper bound set by the faster (80 beat/sec) CPG. Despite early evolvability leading to a statistically significant advantage in the first 150 generations, in later generations the touch sensor setup is unable to produce creatures that travel as far as the faster CPG setup $(p=1.27\times 10^{-7} {\rm \ at\ Gen.\ 1000})$. Artificially throttled, the slow CPG is unable to compete with either $(p<10^{-16})$.

Expansion/Contraction Cycle

In the soft robot evolution system described by Cheney et al. (2013), regular, quickly repeating, and coupled out-of-phase sinusoidal action cycles defined the expansion and contraction of cells. In this model, which does not feature the same complimentary muscle types, the question of actuation cycle is not entirely clear. In an attempt to explore this, here we test the effectiveness of contraction-then-expansion phase cycles against expansion-then-contraction cycles (Fig. 9). These treatments take place on the baseline (slow) pacemaker setup, as to not allow continuous and quickly repeating expansion/contraction cycles, but rather to have a break between actuations. Despite the same number of beats (and thus the same amount of overall expansion and contraction) in both setups, the contraction-then-expansion setup performs significantly better $(p = 1.94 \times 10^{-3})$. While the reason for this difference is not entirely clear, it may be due, in part, to a larger continuous expansion period from the trough of the sine wave to its peak (continuous expansion from minimum to maximum size) in the contraction-then-expansion treatment. In contrast, the expansion-then-contraction setup includes a full-cycle length pause in the middle of its expansionary period. This explanation would suggest that more locomotion tends to come from pushing than pulling, which is in line with our observations from viewing videos of the evolved behaviors.

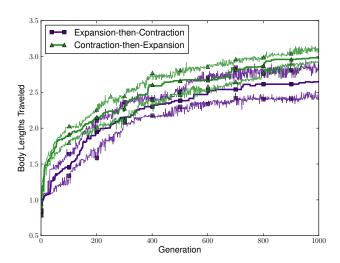


Figure 9: Unlike the regularly occurring actuation cycles of Cheney et al. (2013), the electrophysiological actuations in this paper do not have a necessary order: either expansion or contraction can occur first. It turns out that performance is significantly higher when muscles contract first and then expand, rather than vice versa $(p = 1.94 \times 10^{-3})$.

Discussion

This work reduces the separation between bodies and brains in research into embodied cognition. We did so by embedding the control systems into the physical simulation of the robot's morphology. Perhaps most interesting about this work is that the complex and interesting behaviors are the direct result of the morphology of the creatures, as the control is woven directly into the structure of the organisms. In this work the size of the creatures was limited for computational reasons, but in future work we plan to explore larger design spaces. We also plan to test different ways of implementing electrophysiological robots and to challenge them to perform more difficult tasks.

Conclusion

We have introduced electrophysiological robots, which are inspired by the electrical properties of cardiac tissue. The behavior of these robots is governed by electrical signals flowing though the evolved cells of soft robots. We described these robots and how they are evolved, including the evolution of interesting behaviors, despite the added challenge of physically embedded control structures. We also provided an initial experimental investigation into different implementation decisions, such as alternatives for sensing, actuation, and central pattern generator locations. We believe that this paper provides a first step towards removing the gulf between brains and bodies to encourage further research into physically realistic embodied cognition.

References

- Auerbach, J. and Bongard, J. C. (2009). How robot morphology and training order affect the learning of multiple behaviors. In *IEEE Congress on Evol. Comp.*, 2009, pages 39–46. IEEE.
- Auerbach, J. E. and Bongard, J. C. (2010a). Dynamic resolution in the co-evolution of morphology and control. In *ALife XII*.
- Auerbach, J. E. and Bongard, J. C. (2010b). Evolving CPPNs to grow three-dimensional physical structures. In *Proc. of the Genetic & Evolutionary Comp. Conf.*, pages 627–634. ACM.
- Auerbach, J. E. and Bongard, J. C. (2012). On the relationship between environmental and morphological complexity in evolved robots. In *Proc. of Genetic & Evol. Comp. Conf.*, pages 521–8. ACM.
- Auerbach, J. E. and Bongard, J. C. (2014). Environmental influence on the evolution of morphological complexity in machines. *PLoS computational biology*, 10(1):e1003399.
- Belter, D., Kasinski, A., and Skrzypczynski, P. (2008). Evolving feasible gaits for a hexapod robot by reducing the space of possible solutions. In *Intelligent Robots and Systems*, 2008. *IROS 2008. IEEE/RSJ International Conference on*, pages 2673–2678. IEEE.
- Bongard, J. C. (2013). Evolutionary robotics. *Comm. of the ACM*, 56(8):74–83.
- Brown, H. F. (1982). Electrophysiology of the sinoatrial node. *Physiological Reviews*, 62(2):505–530.
- Cheney, N., MacCurdy, R., Clune, J., and Lipson, H. (2013). Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. In *Proc. of the 15th Genetic and Evol. Comp. Conf.*, pages 167–174. ACM.
- Clune, J., Beckmann, B., Ofria, C., and Pennock, R. (2009). Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *Proc. of the IEEE Congress on Evol. Comp.*, pages 2764–71.
- Clune, J., Stanley, K. O., Pennock, R. T., and Ofria, C. (2011). On the performance of indirect encoding across the continuum of regularity. *IEEE Trans. on Evol. Comp.*, 15(4):346–67.
- Fenton, F. H., Cherry, E. M., Karma, A., and Rappel, W.-J. (2005). Modeling wave propagation in realistic heart geometries using the phase-field method. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 15(1):013502.
- Floreano, D. and Mattiussi, C. (2008). *Bio-inspired artificial intelligence: theories, methods, and technologies*. MIT Press.
- Hiller, J. and Lipson, H. (2014). Dynamic simulation of soft multimaterial 3d-printed objects. *Soft Robotics*, 1(1):88–101.
- Hiller, J. D. and Lipson, H. (2012a). Automatic design and manufacture of soft robots. *IEEE Trans. on Rob.*, 28(2):457–466.
- Hiller, J. D. and Lipson, H. (2012b). Dynamic simulation of soft heterogeneous objects. *ArXiv:1212.2845*.
- Hoffman, B. F., Cranefield, P. F., and Johnston, F. D. (1960). Electrophysiology of the heart.

- Hornby, G. S., Pollack, J. B., et al. (2001). Body-brain co-evolution using l-systems as a generative encoding. In *Proc. of the Genetic and Evolutionary Comp. Conf.*, pages 868–875.
- Ijspeert, A. J., Crespi, A., Ryczko, D., and Cabelguen, J. M. (2007). From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315(5817):1416–1420.
- Lehman, J. and Stanley, K. O. (2011). Evolving a diversity of virtual creatures through novelty search and local comp. In *Proc.* of 13th Genetic & Evol. Comp. Conf., pages 211–8. ACM.
- Lipson, H. and Pollack, J. B. (2000). Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974–978.
- McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of mathematical biology*, 5(4):115–133.
- Pfeifer, R. and Bongard, J. C. (2006). How the body shapes the way we think: a new view of intelligence. MIT press.
- Pfeifer, R., Lungarella, M., and Iida, F. (2007). Self-organization, embodiment, and biologically inspired robotics. *science*, 318(5853):1088–1093.
- Rieffel, J., Knox, D., Smith, S., and Trimmer, B. (2013). Growing and evolving soft robots. *Artificial Life*, (Early Access):1–20.
- Segev, I. and Schneidman, E. (1999). Axons as computing devices: basic insights gained from models. *Journal of Physiology-Paris*, 93(4):263–270.
- Sims, K. (1994). Evolving 3d morphology and behavior by competition. *Artificial life*, 1(4):353–372.
- Sokal, R. and Rohlf, F. (1995). Biometry: the principles and practice of statistics in biological research. WH Freeman.
- Stanley, K. O. (2006). Exploiting regularity without development. In *Proceedings of the AAAI Fall Symposium on Developmental Systems*, page 37. AAAI Press Menlo Park, CA.
- Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines*, 8(2):131–162.
- Zullo, L., Sumbre, G., Agnisola, C., Flash, T., and Hochner, B. (2009). Nonsomatotopic organization of the higher motor centers in octopus. *Current Biology*, 19(19):1632–1636.

A model-based framework to investigate *morphological computation* in *muscular hydrostats* and to design soft robotic arms

Vito Cacucciolo, Matteo Cianchetti and Cecilia Laschi The BioRobotics Institute, Scuola Superiore Sant'Anna, Pisa, Italy { v.cacucciolo, m.cianchetti, c.laschi } @sssup.it

Extended Abstract

Soft Robotics is basically intended as building robots with highly compliant materials, but it is indeed more. Soft robots can safely interact with humans and with the environment and be able to adapt to different situations. These characteristics, combined with cheap materials and simple fabrication, candidate them to lead the next robotics revolution, when robots will massively move from the highly controlled industrial environments to the unpredictable real ones. As for the hardware, compliant materials such as silicones substitute stiff metals and rigid joints, following in under-actuated robots with virtually infinite number of degrees of freedom (DOF).

Controlling this extremely high dexterity using the same approach of hard robotics (i.e. a hierarchical top-down control) simply does not apply. On the contrary, a new vision of the coupling between body and intelligence has to be adopted. Instead of considering brain and morphology separated entities (with the first commanding and the second obeying) *morphological computation* (Pfeifer and Bongard, 2007) proposes a radically different standpoint. The nonlinearity and complexity of the body dynamics rather than problems are considered as part of the solution, as the produced richness of behaviors allow the morphology to carry part of the computational load, thus simplifying the required (central) control instead of complicating it.

This phenomenon is well observed in self-organization and embodiment of biological organisms, where characteristics like adaptability, robustness and agility are widely present without being explicitly controlled, and often without any (centralized) control. In the recent years many examples of embodiment in complex biological systems have been source of inspiration for robotics, ranging from cellular reproduction to the locomotion of more evolved animals like the salamander. A remarkable case where a central brain is present but still most of the sensory-motor control emerges from the body dynamics and the interaction with the environment is the octopus.

The octopus (octopus vulgaris) is an invertebrate sea animal showing high dexterity, variable stiffness and much

more complex behaviors than expected from its position in the evolutionary scale. Its body has virtually infinite DOF, resulting in a very high computational cost if pretended to be fully controlled by its central nervous system. It has been proved instead that its relatively simple brain mainly sends actuation patterns and most of the computation is performed by a combination of the peripheral nervous system, the dynamics of the compliant body and the interaction with the environment (Yekutieli et al., 2005). The result is an underactuated embodied system.

The element at the base of this system is the *muscular hydrostat*. It is an isovolumetric structure widely present in nature as a component of compliant animal bodies. Besides generating the forces for movements it also provides skeletal support, featuring extremely high dexterity and variable stiffness ability. It is composed by a combination of muscular fibers arranged in longitudinal, transverse and oblique directions (Fig. 1). The different activation patterns of these

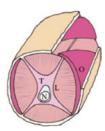


Figure 1: Longitudinal (L), transverse (T), oblique (O) muscular fibers and nerve cord (N) in muscular hydrostats.

muscular fibers generate all the basic movements performed by muscular hydrostats. The whole octopus body can indeed be considered as a complex system where the interaction above its elements and the environment lead to the emergence of the global behavior. The brain orients its outcome rather then controlling each intermediate step. Beyond being widely studied in biology, the octopus represents a rich source of inspiration for roboticists (Laschi et al., 2009), who want to reproduce its peculiar characteristics to build marine soft robots able to swim, manipulate, and move in

unstructured environments.

Understanding the principles underlying the behaviors of the octopus (and of other similar natural systems) is thus not only interesting from a biological standpoint but extremely useful for the design of a new generation of embodied soft robots. So far, this design has been mainly founded on biomimetics, meaning trying to reproduce biological structures in order to take advantage of their qualities. While this method has proven to be a powerful tool, it has two main shortcomings: the technological limits often encountered in reproducing biological structures (e.g., artificial muscles) and the challenge of applying embodiment principles to perform a defined task.

This reveals the need for a systematic design framework able to extract the basic principles of the embodied intelligence, transferring them into an artificial system while taking into account the technological limits and the (main) tasks to perform. In order to apply this *Design for Emergence* (i.e. shape the morphology so that the desired behaviors are likely to emerge) a modeling phase has to be included, where the complex characteristics of the system are synthesized and quantified and the technological constraints included.

In this work a lumped-parameters mathematical model of the nonlinear dynamics of a muscular hydrostat is proposed, inspired to the work of Yekutieli et al. (2005), where a similar model has been used to study the octopus reaching behavior. The presented model is realized as a complex network of masses and springs in order to reproduce the rich dynamics of the octopus arms (Fig. 2). Furthermore it is general, thus

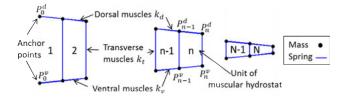


Figure 2: Scheme of the lumped-parameter arm model.

able to simulate a wide range of animal structures based on muscular hydrostats (e.g., octopus arm, elephant trunk) as well as, fittingly setting the parameters, soft robotic arms with muscle-like actuation (e.g., pneumatic, hydraulic).

Differently from other similar models, the equations of motion were obtained using an energy-based method (*Lagrangian mechanics*). The two main reasons are: have a set of equations where it is easy to add various constraints; energy and generalized coordinates provide a full description of the evolution of the system, the former at a global level, the latter at a local (*agent*) one. Constraints were applied to add the isovolumetric feature of muscular hydrostats and contacts (a novelty in these studies) to model the interaction with the environment (Fig. 3), opening to the investigation of grasping tasks and safe interaction with humans.

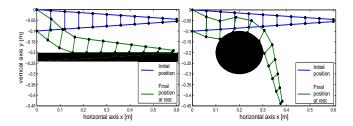


Figure 3: Soft arm free falling in contact with a hard plane (left) and sphere (right).

The proposed model is meant to be combined with evolutionary computation tools in a set-up for brain-body co-evolution (Lipson and Pollack, 2000): a genetic algorithm contemporary optimizing a neural controller (the brain) and the arm model parameters (the body), with the fitness mainly represented by the performance on an assigned task (e.g., reaching a point, grasping an object). Respect to other similar works the evolution is not from *tabula rasa*, as the design space, contemplating the different combinations of morphology and control, is extremely wide and with plenty of local minima. Thus, in order to simplify the genetic optimization process, bio-inspiration is used to focus the design space about biological systems well-suited for the desired tasks.

The aim of this set-up is twofold. First, the quantitative and task-related analysis of embodied intelligence and morphological computation in biological systems based on muscular hydrostats: thanks to the realized coupling the body parameters influence how the control is shaped by the genetic optimization and *vice versa*. Second, setting the model parameters to fit technological constraints (relative to a chosen actuation technology), the definition of a new framework for the Design for Emergence of soft robotic arms, where morphology and control are simultaneously optimized to reach the emergence of the desired global behaviors with the simplest software and hardware.

Acknowledgements

This work is supported by RoboSoft - A Coordination Action for Soft Robotics (FP7-ICT-2013-C # 619319).

References

Laschi, C., Mazzolai, B., Mattoli, V., Cianchetti, M., and Dario, P. (2009). Design of a biomimetic robotic octopus arm. *Bioinspiration & Biomimetics*, 4(1):015006.

Lipson, H. and Pollack, J. B. (2000). Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974–978.

Pfeifer, R. and Bongard, J. (2007). How the body shapes the way we think: a new view of intelligence. MIT press.

Yekutieli, Y., Sagiv-Zohar, R., Aharonov, R., Engel, Y., Hochner, B., and Flash, T. (2005). Dynamic model of the octopus arm. i. biomechanics of the octopus reaching movement. *Journal of neurophysiology*, 94(2):1443–1458.

Evolution of Locomotion on a Physical Tensegrity Robot

Mark Khazanov¹, Jules Jocque¹, and John Rieffel¹

¹Computer Science Department, Union College, Schenectady NY 12308 rieffelj@union.edu

Abstract

Due to their high strength-to-weight ratio, robustness and deformability, tensegrity structures are an appealing platform for the emerging field of soft robotics, with applications ranging from search-and-rescue to field-deployable structures. Unfortunately, these properties also make tensegrities challenging to control through conventional means. In this paper we describe a novel means of vibration-based tensegrity actuation which allows for the manual control of a physical tensegrity robot in the plane as well as state-machine based target following. We demonstrate the evolution of effective gaits using only physical evaluations of the robot, and further demonstrate how a combination of the state-machine with the hill climber allows for the hands-off automation of the evolutionary process. We conclude with a description of how this can lead to a bootstrapping effect, with the potential to significantly accelerate and automate the physical evolution of our tensegrity robot.

Introduction

The term *tensegrity* describes a class of structures which are held together through *pre-stress* - a synergistic interplay of compressive and tensile forces. In engineering and architecture, tensegrity structures ranging in scale from camping tents to large stadium roofs are commonly built out of little more than compressive struts and tensile strings.

Because of their resilience and deformability, tensegrity structures are an appealing robotics platform. A tensegrity robot can alter its shape while preserving its structural stability merely by changing the resting length of its strings. Unfortunately, this property of *pre-stress* also adds a significant amount of dynamical complexity to the system. Because local changes to tension are redistributed throughout the structure, the system is dynamically coupled and highly resonant (Bohm et al., 2014) This dynamical complexity presents a challenge when trying to control tensegrities through conventional means.

Interestingly enough, the principle of tensegrity also exists throughout the biological world, at scales ranging from the musculoskeletal system of animals to the arrangements of the cellular cytoskeleton. The field of *morphological computation* (Paul, 2006; Pfeifer and Bongard, 2007) seeks

to explore how dynamical complexity can sometimes be advantageous in both biological and engineering realms, and how the intelligent behavior of systems can sometimes be *hard-wired* into their mechanics.

Tensegrities are therefore a compelling choice for exploring morphological computation. Broadly, our research explores the paradoxical notion that increasing a system's dynamical complexity can sometimes simplify its control. Specifically, we are interested in understanding how the dynamical coupling of tensegrity-based robots can be exploited as an advantage.

In this paper we demonstrate how the locomotion of a tensegrity robot can be controlled in the plane simply by changing the frequency of three vibrational motors. This steering behavior can then be combined with a finite state machine, allowing for autonomous target tracking. Moreover, we show how the motor frequencies can be optimized via evolutionary techniques, producing increasingly effective gaits. We believe that this is the first example of evolution on a physical tensegrity robot, and is the smallest and the fastest physical tensegrity robot in its class.

Understanding how to make dynamically complex tensegrities move will not only allow us not only to contribute to the emerging field of soft robotics, it will also add to our understanding of the biological systems which inspire us.

Tensegrity Robots

Conventionally, robotic systems are controlled by first dampening their dynamic and vibrational modes, and then controlling them in the quasi-static realm, in which analytical methods like inverse kinematics are capable of producing reliable controllers. The same is true of conventional approaches to the control of tensegrity structures. Skelton *et al.*, for instance, have been able to demonstrate both active vibration damping (2004) and open-loop control of simple structures (2004). Once this resonance is tamed, the robots are moved by changing the rest lengths of the tensile elements, for instance via servo motors (Paul et al., 2006, 2005)

Because of the challenge in building and controlling phys-

ical tensegrity robots, the majority of tensegrity robot research has occurred in simulation rather than reality (Aldrich et al., 2003; Paul et al., 2006; Graells Rovira and Mirats Tur, 2009; Iscen et al., 2013). One notable recent contribution is Caluwaerts *et al.*'s work (2013) on physical reservoir computing in a simulated tensegrity robot, in which they demonstrate learnable gaits produced by relatively simple central pattern generators (CPGs). More recently, Iscen *et al.* (2013) have evolved rolling gaits on a simulated tensegrity robot.

The case for physical evolution

While these simulated results are promising, the dynamics of tensegrities are complex enough that there is little hope of reliably breaching the *reality gap* (Jakobi et al., 1995) between simulated and actual dynamics. Not surprisingly, there are very few published examples of physically embodied tensegrity robots moving.

Shibata *et al.* (2009) and later Koizumi *et al.* (2012) built six-bar robots capable of motion. In each case, the controller was discovered manually. More relevant to our work, Paul *et al.* (2006) evolved gaits for a 3-bar tensegrity in simulation and then transferred the controller to a physical robot with only minor success.

Most of the tensegrity robot controllers described above operate at slow speeds relative to the structural dynamics, and can therefore be considered to be quasi-static – none of them truly take advantage of the inherent resonance of tensegrity structures. A notable recent exception is Zimmermann *et al.* (2011), who, like us, used vibration of a single motor as a basis for locomotion, however theirs is a modified (class-II) tensegrity, and only capable of forward motion.

The dynamics of tensegrity robots are sufficiently complex that there are no known purely analytical methods of generating gaits. As a consequence, every gait described above (with the exception of Paul's) was optimized through human trial-and-error, at the expense of considerable time and effort. This makes the domain particularly appealing for automated evolutionary approaches, which can remove human interaction (which can lead to error and bias in the search) from the discovery of effective gaits.

Acknowledging the limits of simulation and transferability however, limits us to *physical* rather than simulated evolution of tensegrity locomotion. This research then falls under the rubric of the Evolution of Physical Systems (sometimes called embodied evolution), following in the footsteps of Harvey *et al.* at Sussex (Harvey et al., 1997), Watson *et al.* (Watson et al., 1999), and more recently Zykov (2004) and Yosinksi (2011).

Unlike simulated evolution, the evolution of physical systems can be painstakingly slow, and requires considerable human interaction. Also unlike simulated evolution, however, our evolved solutions are, by fiat, guaranteed to work in the real world. As Rodney Brooks claimed, "the world is

its own best model" (1990).

A Robot Designed to Resonate

The motivation for our work lies in striving to exploit, rather than suppress, this inherent dynamical complexity as an advantage – making tensegrities move by vibrating, rather than suppressing their vibrations. Exploring these ideas required building a physical tensegrity robot.

Our ambition was to design a small tensegrity that was powered by vibration alone, robust enough to withstand prolonged testing and easy to manufacture and repair. The resulting design, based upon a canonical six-bar tensegrity shape, is shown in Figure 1. The geometry is defined by six equal length composite struts which are connected to each other via 24 identical helical springs, with four springs emanating from each strut end. Our prior work Khazanov et al. (2013) describes the design choices of the robot in more detail.

The robot is actuated with three independent vibrational motors capable of operating between 100 and 260 Hz. Each motor was mounted on a separate strut (leaving three of the six struts empty). Each motor is independently controlled by an off-board USB motor controller connected to a host computer. Very thin wires were used to tether the motors to the controllers, allowing for a minimal amount of interference through drag. Enough tether wire was used to enable a significant amount of twisting (caused by rotation of the tensegrity) before the need to manually unwind the tether.

The motion of the tensegrity robot could then be determined simply by independently varying the three motor frequencies. An analysis of the complexity of the ensuing dynamics can be found in our earlier work Khazanov et al. (2013).

The sections below describe various approaches to finding motor frequency sets capable of producing interesting locomotive gaits on our designed robot.

Setup

In order to evaluate gaits, the robot was placed on a 91x61 cm table (Figure 2) with a removable cork-board sheet as the surface. This material was chosen in order to provide enough friction for the tensegrity to move freely, while also allowing for it to grasp the surface. Wooden borders surrounded the testing environment to guard the tensegrity from falling during trials.

As illustrated by Figure 3, the process of distance measure was automated using an overhead USB camera connected to the control computer. The location of the tensegrity in a frame was determined by subtracting the image of an "empty" arena from an image containing the robot and then finding the centroid of the remaining pixels. The tether is visible in some frames, but has a negligible impact upon positional measurements. Distance could then be calculated by comparing the pre- and post-evaluation locations. The

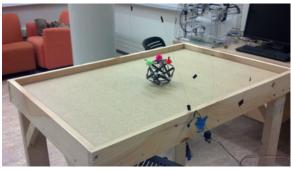




Figure 2: Left: The testing arena was a 91x61 cm table with a cork-board floor and raised wooden walls. The tensegrity robot was tethered to the control computer using thin wires. Right: A closeup of the tensegrity with the colored tracking markers. The yellow ball in the distance is the object used for target pursuit.



Figure 1: Our tensegrity robot is actuated by three independent pager motors mounted directly to the struts. Each motor is connected to an off-board USB motor controller with a very thin wire. (Photo by Steven Stangle)

arena was large enough that multiple evaluations could often be performed before manually returning the robot to the center of the arena.

Evolution of Motion

Using this setup, we then used a series of approaches to discover effective and interesting locomotive gaits for the robot. In each approach, the behavior of the robot was determined entirely by a "genotype" or frequency set, consisting of three motor voltages corresponding to frequencies for the three vibrational motors on the robot.

Hand Selected Gaits

The first step in our process was to use an interactive process to discover interesting gaits via human trial and error. A simple python-based user interface was set up allowing an operator to vary motor voltages while observing the behavior of the robot. Three particularly interesting gaits were discovered: one which produced linear "forward" motion

(the robot has no pre-defined front or back), one which rotated the robot clockwise, and one which rotated the robot counter-clockwise.

We could then manually switch between these three frequency sets in order to "steer" the robot around the arena (Khazanov et al., 2013). A video of this behavior can be found on our web page (http://cs.union.edu/~rieffelj/videos.html)

Target Pursuit in the 2-D plane

We then implemented a finite state machine which incorporated the three gaits described above in order to achieve fully automated target tracking.

To begin, we developed a machine vision algorithm, written in OpenCV, capable of using the overhead USB camera to detect the tensegrity's orientation on the plane. To do this we placed uniquely colored tracking markers on the three top-most strut ends. Using simple color detection, our algorithm could then detect each marker, and subsequently compute their center of mass along with the relative vector between the center and each colored marker. The lower row of images in Figure 3 illustrate this. The pink ball represents the "rear" of the robot - meaning that the vector from the robot's center of mass toward the front points in the opposite direction – as illustrated by the short line radiating from the center of mass.

A yellow target marker was then placed in the arena, and a steering angle calculated as the difference between the "forward" vector (the short line in Figure 3) and the vector to the target (the longer line in Figure 3). As illustrated by Figure 4 this steering angle, along with the robot's distance to the target, was then used as input into a simple four-state finite state machine. When the target was in front of the robot $(-30^{\circ} > steeringAngle < 30^{\circ})$, the forward gait was used. When the target was to the right $(30^{\circ} > steeringAngle > 180^{\circ})$, the clockwise gait was used, and when the target was to the left $(-30^{\circ} < steeringAngle < -180^{\circ})$, the counterclockwise gait was used.

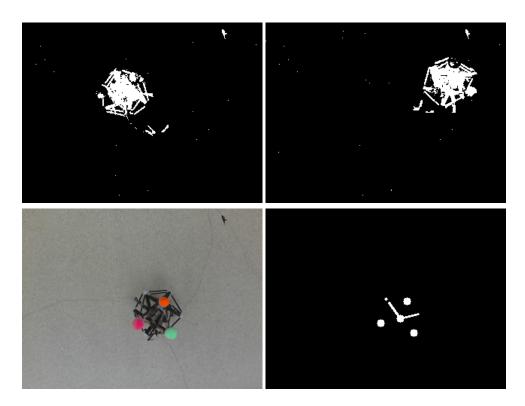


Figure 3: Using images produced by an overhead camera, distance traveled could be determined by measuring the difference in the center of mass between initial (top left) and final (top right) frames. Using colored markers (bottom left), we were then able to use machine vision to calculate the robot's orientation (bottom right). The short line indicates the "forward" vector or the robot. The longer line shows the vector to the target.

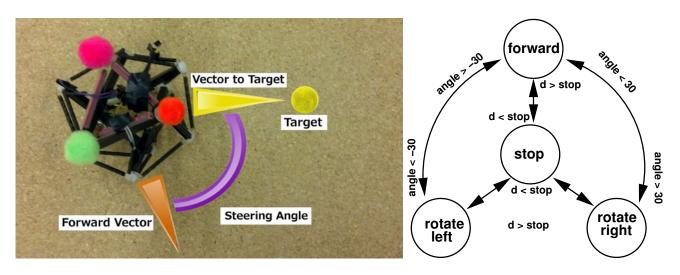


Figure 4: Left: Using machine vision code written in OpenCV, we were able to detect the location of colored markers on the tensegrity and use this information to determine the robot's absolute orientation, as well as relative angle to the target object. Right: this steering angle was then used as input into a finite state machine which switched between forward and rotational gaits.

When the machine vision algorithm, the state machine, and the hand-selected gaits were combined in this manner, it allowed for quite effective target tracking in the 2D plane. A video on our web page shows this target tracking behavior in action.

Evolving Gaits with Human Interaction

Our next step was to automate the trial-and error process used to select effective forward-moving gaits by implementing a simple population-based hill climber. The aim was to find forward moving gaits which outperformed those produced by trial and error in the section above.

The genotype, consisting of three motor frequencies, is insufficiently complex to justify more complex search techniques. (In the future we will be moving toward more evolvable representations and controllers, such as Artificial Neural Networks).

To perform the search we used a population size of 10 individuals, with 50% elitism (keeping the best half of the population between generations). In order to account for the noisiness of real-world evaluation, during successive generations all members of the population were evaluated, including those that had been carried over from prior populations.

Evaluation of a motor frequency set genotype was performed by measuring linear distance traveled by the tensegrity over the course of 7 seconds. The distance was measured using the automated machine vision algorithm described above.

This evolutionary process produced results, but was incredibly time intensive, since human effort was required to manually reset the tensegrity back to the starting position in the arena between evaluations. This inspired us to consider a more hands-off approach.

Hands-free Evolution of Gaits

Combining the target-following state machine described in an earlier section with the hill climber described in the previous section allows for the hands-free evolution of gaits.

This is accomplished by using the state machine to steer the tensegrity back to the starting position after every evaluation. In effect, this was as simple as treating the starting position (always the center of the arena frame) as a virtual target for the state machine's steering vector calculation.

This hands-free evolution allowed a significant number of repeated trials without any human interference at all, and an overall reduction in total number of interactions during an evolutionary run. The following section provides a more quantitative and comparative analysis of interactions required between the two setups.

Nonetheless, the physical evolution of tensegrity robots is still time intensive (if less labor intensive). Figure 5 shows maximum fitness achieved over the course of 25 generations between two different evolutionary runs. The non-monotonically increasing nature of the search is due to the

fact that all members of the population were re-evaluated. Evaluation of tensegrity robots is an inherently noisy process because of a variety of complex interactions and sensitivities such as friction, motor phase and motor hysteresis. Our discussion below explores better ways to approach these issues.

Results and Discussion

The hands-on and hands-free approaches to evolution produced equivalent fitnesses over 25 generations, and we make no claims that one produces quantitatively better gaits than the other. Nor would one necessarily expect otherwise, since the only distinction between the two algorithms – how the robot returns to the starting location – has no bearing on fitness evaluations or the larger search technique.

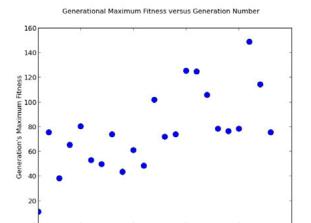
In both cases, since we re-evaluated the entire population each generation, and only used the latest fitness for the purposes of selection, there is not the typical monotonic increase in fitness found in most GAs. In the near term we are interested in more sophisticated measures of dealing with noisy fitness evaluation - for instance by evaluating every genotype multiple times and keeping track of a running average. The field of noisy evaluation provides some insight on the matter (Fitzpatrick and Grefenstette, 1988). Finding elegant solutions to this issue has bearings on the entire field of physical evolution.

Where our two approaches differ most is on the qualitative level. The hands-on approach requires, at a minimum, one human interaction every 7 seconds in order to return the robot to the start position, meaning a 25-generation run with population size 10 involves 250 such interactions. (We are reminded of anecdotes of Zykov *et al's* (2004) physical evolution of a massive 9kg pneumatically actuated Nonaped, which had to be moved 3m back to its starting position between trials)

In contrast to the hands-on approach, our hands-off approach at best requires 0 human interactions during a 25-generation run, because the robot is able to steer itself back to the starting position.

In practice, the number of interactions required in both approaches was higher, due to the inevitable wear-and-tear on the hardware caused by repeated trials, often manifesting itself in broken soldering at the wire connection points. A second phenomenon which required human intervention was the twisting of the tether caused by too many repeated turns in one direction. There was no significant difference in the number of repair-related interventions between the two algorithms.

Our choices of a very simple genetic encodings (three numbers) and a very simple algorithm (a hillclimber) are deliberate. Our aims have been to demonstrate the feasiblity of the physical evolution of tensegrity gaits, to produce effective gaits, and to introduce a novel method of reducing the amount of human interaction required for the evolution of



10 15 Generation Number



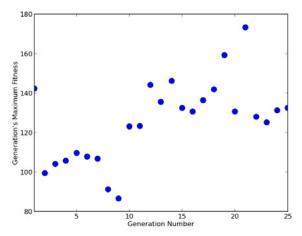


Figure 5: Fitness data from two consecutive 25-generations hands-free evolutionary runs of the tensegrity robot. Fitness measures are in cm of displacement over the course of a 7 second evaluation. The non-monotonically-increasing nature of the results is due to the re-evaluation of generational bests.

physical systems (by having the robot automatically reset itself between evaluations). More elaborate approaches might have distracted from these points. There are certainly more sophisticated algorithms for the Evolution of Physical Systems which we are eager to explore in the near future - especially the (1+1) restart-online adaptation algorithm (Bredeche et al., 2010; Montanier and Bredeche, 2011).

There are several modifications we could make at the hardware and software level to improve the robustness of the system and reduce the amount of intervention required. At the hardware level, we are exploring more sophisticated tether designs, such as those involving ring couplings. In the longer term, we are interested in transitioning to a completely wireless setup, with rechargeable batteries. On the software front, it would be relatively simple to keep track of the cumulative rotation of the robot over the course of evolution, and automatically "unwind" the tether when the sum crossed a threshold.

Toward Bootstrapping Evolution of Gaits

The most significant innovation to our algorithm lies ahead. We want to improve the hands-off method by more tightly coupling the state machine (used to return the robot to the starting location) with the learning which takes place in the evolutionary search for forward moving gaits. We envision a means by which, as newer faster forward gaits are discovered by the GA they could be used to "upgrade" the go-forward behavior of the state machine. This bootstrapping would in principle lead to increasingly shorter times between evaluations and therefore an overall increase in experimental throughput. We could even integrate aspects of novelty search (Lehman and Stanley, 2008).

The evolution of physical, rather than simulated systems,

poses many intriguing challenges, but offers the promise of finding novel solutions which immediately work in the real-world, thereby avoiding Jakobi's dreaded "reality gap", at the cost of real-world time and human interaction. Any innovation which is able to decrease these costs is therefore a valuable contribution to the field. After all, unlike simulations, one can't speed up the real world by buying a faster computer.

Conclusion

We have described a sequence of approaches to the discovery of novel vibration-based locomotive gaits in a physically embodied tensegrity robot. To the best of our knowledge, this is the first example of the physical evolution of tensegrity robot locomotion. Our discovered gaits operate fully in the dynamic range, and are successful largely because they are able to exploit, rather than suppress the inherent dynamical complexity of tensegrity structures. Broadly, these results lend further credence to the field of *morphological computation* and the notion that dynamical complexity in both living and engineered systems can sometimes be an advantage – minimizing the overall cost of control by "outsourcing" intelligence directly into the mechanics of a structure.

Acknowledgments

The authors would like to thank Steve Stangle and Kadeam Vendreyes who assisted in this research as part of undergraduate practica at Union College. Funding for undergraduate summer research by authors Mark Khazanov and Jules Jocque was provided through the generosity of the Union College Summer Research Fellowship Program. Materials

were paid for in part by several Union College Internal Education Fund (IEF) and Faculty Research Fund (FRF) grants to the authors.

References

- Aldrich, J., Skelton, R., and Kreutz-Delgado, K. (2003). Control synthesis for a class of light and agile robotic tensegrity structures. In *American Control Conference*, 2003. Proceedings of the 2003, volume 6, pages 5245–5251. IEEE.
- Bohm, V., Zeidis, I., and Zimmermann, K. (2014). An approach to the dynamics and control of a planar tensegrity structure with application in locomotion systems. *International Journal of Dynamics and Control*, pages 1–9.
- Bredeche, N., Haasdijk, E., and Eiben, A. (2010). On-line, on-board evolution of robot controllers. In *Artifical Evolution*, pages 110–121. Springer.
- Brooks, R. A. (1990). Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15.
- Caluwaerts, K., D'Haene, M., Verstraeten, D., and Schrauwen, B. (2013). Locomotion without a brain: Physical reservoir computing in tensegrity structures. *Artificial life*, 19(1):35–66.
- Chan, W. L., Arbelaez, D., Bossens, F., and Skelton, R. E. (2004). Active vibration control of a three-stage tensegrity structure. In Proceedings of SPIE 11th Annual International Symposium on Smart Structures and Materials.
- Fitzpatrick, J. M. and Grefenstette, J. J. (1988). Genetic algorithms in noisy environments. *Machine learning*, 3(2-3):101–120.
- Graells Rovira, A. and Mirats Tur, J. M. (2009). Control and simulation of a tensegrity-based mobile robot. *Robotics and Autonomous Systems*, 57(5):526–535.
- Harvey, I., Husbands, P., Cliff, D., Thompson, A., and Jakobi, N. (1997). Evolutionary robotics: the sussex approach. *Robotics and autonomous systems*, 20(2):205–224.
- Iscen, A., Agogino, A., SunSpiral, V., and Tumer, K. (2013). Controlling tensegrity robots through evolution. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*, pages 1293–1300. ACM.
- Jakobi, N., Husbands, P., and Harvey, I. (1995). Noise and the reality gap: The use of simulation in evolutionary robotics. In *Proc. of the Third European Conference on Artificial Life* (ECAL'95), pages 704–720, Granada, Spain.
- Khazanov, M., Humphreys, B., Keat, W., and Rieffel, J. (2013). Exploiting dynamical complexity in a physical tensegrity robot to achieve locomotion. In *Advances in Artificial Life*, *ECAL*, volume 12, pages 965–972.
- Koizumi, Y., Shibata, M., and Hirai, S. (2012). Rolling tensegrity driven by pneumatic soft actuators. In *Robotics and Automa*tion (ICRA), 2012 IEEE International Conference on, pages 1988–1993. IEEE.
- Lehman, J. and Stanley, K. O. (2008). Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pages 329–336.

- Masic, M. and Skelton, R. E. (2004). Open-loop control of class-2 tensegrity towers. In *Proceedings of SPIE 11th Annual International Symposium on Smart Structures and Materials*.
- Montanier, J.-M. and Bredeche, N. (2011). Embedded evolutionary robotics: The (1+ 1)-restart-online adaptation algorithm. In *New Horizons in Evolutionary Robotics*, pages 155–169. Springer.
- Paul, C. (2006). Morphological computation: A basis for the analysis of morphology and control requirements. *Robotics and Autonomous Systems*, 54.
- Paul, C., Valero-Cuevas, F. J., and Lipson, H. (2005). Redundancy in the control of robots with highly coupled mechanical structures. In *Int. Conf. on Intelligent Robots and Systems*, pages 802–808.
- Paul, C., Valero-Cuevas, F. J., and Lipson, H. (2006). Design and control of tensegrity robots for locomotion. *IEEE Transac*tions on Robotics, 22(5).
- Pfeifer, R. and Bongard, J. (2007). How the body shapes the way we think: a new view of intelligence. MIT press.
- Shibata, M., Saijyo, F., and Hirai, S. (2009). Crawling by body deformation of tensegrity structure robots. In *Robotics and Automation*, 2009. ICRA'09. IEEE International Conference on, pages 4375–4380. IEEE.
- Watson, R. A., Ficici, S. G., and Pollack, J. B. (1999). Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., and Zalzala, A., editors, *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 335–342, Mayflower Hotel, Washington D.C., USA. IEEE Press.
- Yosinski, J., Clune, J., Hidalgo, D., Nguyen, S., Zagal, J., and Lipson, H. (2011). Evolving robot gaits in hardware: the hyperneat generative encoding vs. parameter optimization. In *Proceedings of the 20th European Conference on Artificial Life, Paris, France*, volume 8, page 12.
- Zimmermann, K., Bohm, V., and Zeidis, I. (2011). Vibration-driven mobile robots based on magneto-sensitive elastomers. In Advanced Intelligent Mechatronics (AIM), 2011 IEEE/ASME International Conference on, pages 730–735. IEEE.
- Zykov, V., Bongard, J., and Lipson, H. (2004). Evolving dynamic gaits on a physical robot. In *Proceedings of Genetic and Evolutionary Computation Conference, Late Breaking Paper, GECCO'04*.

Fine grained artificial development for body-controller coevolution of soft-bodied animats

Michał Joachimczak^{1,2}, Reiji Suzuki¹, Takaya Arita¹

Graduate School of Information Science, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan Systems Modeling Laboratory, Institute of Oceanology, Polish Academy of Sciences, Sopot, Poland mjoach@alife.cs.is.nagoya-u.ac.jp

Abstract

We apply the concept of multicellular development to evolve bodies and controllers of soft-bodied animats, evaluated within simulated physics environment. Growth of an embryo is driven by a neural network-like controller in every cell. We show how, through an evolutionary process, this simple approach produces rich and complex morphologies of animats made of hundreds of cells and exhibiting life-like gaits without centralized control. We discuss design decisions and issues that needed addressing to make evolution of moving bodies possible and investigate how some of these decisions impact quality of obtained solutions. We then show how the system allows us to evolve animats that optimize both for distance and the use of actuating material. Finally, we demonstrate how it can be used to evolve developmentally plastic individuals, whose genomes encode more than one phenotype, each adapted to a different type of environment.

Introduction

The possibility of harnessing evolvability and scalability of biological development in silico has long generated considerable interest in the field of artificial life. The diversity of multicellular forms produced by natural evolution is breathtaking and so is evolution's effectiveness in an endless struggle to tweak and adjust existing forms in response to continuously changing environment. The central role of development has been recognized already at the times Theory of Evolution was originally formulated, while nowadays we also understand that it is owing to the development that structures consisting of trillions of cells can be encoded in genomes of informational content of merely millions of bits. It is thus only natural that the field of evolutionary computation attempts to tap into some of development's potential as a way of overcoming limited scalability of direct genetic encodings (see, e.g., Cheney et al., 2013, for a recent comparison in the relevant domain).

Historically, artificial development models have been classified into two broad approaches: grammatical and cell chemistry based (Stanley and Miikkulainen, 2003). The former use high level, algorithmic abstractions of development, while the latter explicitly simulate interactions of larger numbers of independent cells. When it comes to

body-controller coevolution of simulated animats, we think it would be fair to say that grammatical approaches (such as used in Sims, 1994; Lipson and Pollack, 2000; Komosinski and Ulatowski, 1999; Pilat et al., 2012) and other high level abstractions of development (such as CPPN based, Cheney et al., 2013; Auerbach and Bongard, 2012) have been immensely successful in producing diverse and life-like animat morphologies, of which many have become staples of the artificial life field. The lower level, fine-grained developmental approaches, where morphology emerges through self assembly, have also met with considerable interest and the capability of such systems to evolve complex shapes or display organized behaviors have been well investigated (see Doursat et al., 2013, for a recent review). Yet, when it comes to evolving moving animats, cell chemistry approaches did not seem to have met with as much interest (although see Schramm et al., 2011; Joachimczak et al., 2013; Bongard and Pfeifer, 2003; Meng et al., 2011; Kowaliw et al., 2004, for some notable exceptions). Certainly, a big factor for the limited success of cell chemistry approaches in this domain are their much higher computational costs. After all, ability to define whole body parts algorithmically, e.g., through parameters of geometric primitives, is much more efficient than assembling them from a set of cells, both computationally and from the point of view of reduced search space.

Nonetheless, despite their higher computational costs, we believe that approaches that embody the very ideas of artificial life in the form of self organizing, decentralized assembly deserve further studies and in particular, attempts to scale them from what we would consider a proof of concept level (few dozens of cells) to producing useful and diverse designs consisting of hundreds and thousands of cells.

The core ideas behind the approach presented in this work are not new: it can be considered a generic, cell-chemistry system, with cellular behavior driven by a neural network-like controller in every cell (see, e.g., Roggen and Federici, 2004 for another example), whereas the method for creating elastic animat bodies from multicellular structures has been previously explored by the first author (Joachimczak et al., 2013). However, since the system employed previously was complex and computationally heavy, scaling it up

beyond a few dozens of cells or applying to more complex tasks proved difficult. In this work we have thus attempted to refine and distill core ideas into a much simpler system and demonstrate how a relatively straightforward, physics based multicellular development allows to evolve complex and diverse morphologies of virtual animats built of hundreds of cells and exhibiting life-like gaits.

The model

The main computational cost of the approach we used previously to evolve multicellular animats was related to the model of gene regulatory network (GRN) in which product concentrations would change in a continuous manner and which would be encoded in DNA-inspired genomes of arbitrary length. Although such GRNs were demonstrated to be quite evolvable (Nicolau et al., 2010; Cussat-Blanc et al., 2012; Joachimczak and Wróbel, 2010), frequent network updates and dense topologies resulting from this type of genetic encoding incurred high computational costs. To allow for evolution of more complex animats, we reapproached the problem with a simpler GRN model and refined physics, yet keeping the defining features of fine-grained, artificial development, so that we could use the system to investigate evolution of development in a biologically relevant manner. Hence, the body and the controller of an animat are the product of (i) progressive assembly of a multicellular structure by subsequent cellular divisions; (ii) all cells share a common, GRN-like controller; (iii) yet perform independent decisions based on their local neighborhood and differing activities of their virtual genes. Animats are grown during developmental stage and evaluated during locomotion stage for their capability to produce gaits through expansion and contraction of body regions, a concept that has recently received increasing attention as physical implementations of such robots have become possible (Shepherd et al., 2011; Hiller and Lipson, 2010).

In the following sections we provide description of the model and design choices, although given short space, limited to the essential aspects of the approach.

Development

Developmental process begins with a single cell and proceeds through subsequent cellular divisions. Cells are controlled by a simple abstraction of artificial GRN in the form of a recurrent neural network in which, to allow for richer regulatory logic, nodes can use different activation functions (from a prespecified set). The behavior of a cell depends both on external signals received by its network and network's temporal state represented as activation of its nodes. Inputs to a cell represent external signals from environment (e.g., morphogens, bias) and outputs are used to interpret actions taken by the cell, such as division, death. We used a signed sigmoid as the default activation function, and limited the range of values on output nodes (even if the function type underwent mutation) to <-1,1>.

Physics of development Development takes place in a continuous 2-D space, where cells are represented as discs and undergo elastic collisions simulated with springs that connect them. Cell's physical state is defined by its position, its velocity, and orientation vector which determines the direction of division. Springs connect only the nearest neighbors and are determined dynamically, as the embryo grows. We use Delaunay triangulation to determine the connectivity between cells and then remove links longer than 150% of cell's diameter (to allow for non-convex shapes). The resting length of a spring is equal to the sum of attached cells' radii. As this simple approach can however produce a disjoint structure (e.g., after cells in a central part of an elongated embryo die out), a spring between a pair of nodes is removed only when some other path between these two nodes can be found. Furthermore, having a lesser update cost, the underlying physics of development is iterated at a rate 10 times higher than the networks controlling cell behavior.

Morphogens Morphogen gradients have long been known to play a fundamental role in developmental process and, in particular, in establishing a basic body plan (Carroll et al., 2004). In the experiments discussed in this paper, we provided cells with direct and indirect positional information. The direct one was in the form of X and Y coordinate directly fed as input to a network. The indirect information would come in the form of virtual maternal morphogens at prespecified positions in the environment. Depending on an experiment, corresponding network inputs would report to a cell either morphogen's concentration (decreasing with distance) or a gradient (direction towards the source of this morphogen). Additionally, to allow for a simple form of cell to cell communication, cells have 2 "morphogen" outputs, whose activations are accessible to their cellular neighbors. Through a 2 corresponding inputs, each cell has access to an averaged value of activation of these outputs in its neighbors.

Cell division and death Whenever cell division occurs, a newly created cell receives a copy of original cell's network, including its state, that is current node activations. From then, any change in the patterns of network activities is the result of symmetry being broken owing to the differences in external signals perceived by each of the cells. Division and death (apoptosis) occur when the activation level of the network's node associated with respective function is found to be above a fixed threshold (0.8). The newly created cell is placed next to the original cell, in the direction determined by orientation vector of the original cell. Its angle is determined by the state of the associated output node at the time of division. As the mechanism controlling orientation and timing of cell division has a major influence on the range of morphologies that can be achieved through development, we discuss two approaches we investigated in the results section.

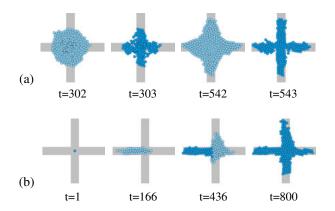


Figure 1: Example snapshots from development of 2 embryos (blue cells) evolved to produce cross shape, overlaid on the target to which embryos would be compared during evaluation (gray). (a) the system allowed for unconstrained cellular divisions whenever an associated output crossed a threshold, the final shape is a result of repeated subtractive process: apoptosis removing excess cells; (b) cell divisions were constrained to non occupied space, evolved shape is a result of an additive assembly. Brightly colored cells have their division signal above the threshold.

Termination of development To prevent trivial scenario in which cells divide in an uncontrolled manner until hard limit of embryo size is reached, we required development to self terminate by stopping both cellular divisions and deaths during last n steps of developmental stage. During evolution, individuals that would not fulfill this criterion would be penalized by having their fitness values divided by a penalty constant. Additionally, we added a limit on the total number of cells that can be created during development of an embryo to be no larger than 4 times the maximum of cells allowed at a time. We did so to limit the occurrence of rather unrealistic solutions we initially observed in which, despite the shape being stable, cells would be indefinitely created and destroyed.

Soft-bodied animat model

The morphology of an embryo in the final developmental step is used as a template for a soft-bodied animat that is then evaluated for its capability to produce gaits. The animat is represented in the physics engine as another springmass system, with point masses located at cell centers and springs forming a triangular mesh. Springs are assigned default resting lengths based on distances between cell centers at the end of developmental stage. Additionally, each triangular region has an equilibrium pressure (also determined at the end of development) and resists excessive compression or stretching. Two examples of transformation from a developed embryo into a soft-bodied animat can be seen in Fig. 4.

Finally, we chose to address a class of solutions that could emerge through developmental process that would be difficult to interpret physically and costly to simulate because of their requirement of increased accuracy to maintain stable behavior, namely solutions in which some of the body regions would have zero thickness. This could happen when a single line of cells would grow out from the main body during development, in which case it would be represented as a chain of point masses attached to the animat. We addressed this scenario by iteratively removing such cells at the end of developmental stage. A related case occurred if two larger body regions were found to be connected through a single cell or a chain of cells (a simplest example of such an individual would be 2 triangles sharing one vertex). We assigned the fitness value of 0 to such animats.

To avoid self penetration of animat bodies, masses representing cells would undergo elastic collisions with springs. Movement during locomotion stage was simulated using a custom soft-body engine built on top of Bullet Physics library. As physics simulation is the most time consuming part of individual's evaluation, we found it highly beneficial to employ a dynamic length of simulation step dt, which would have a larger default value but would decrease on demand, when high accelerations or node-spring collisions were detected.

Locomotion Control is achieved in a distributed manner: each cell independently controls the lengths of its attached springs. After development is finished and locomotion stage begins, network controllers in cells simply continue to function and communicate through morphogens as they did, however, their division/death outputs are now ignored, and the output responsible for actuation can increase or reduce resting lengths of springs attached to a given cell. The resting length of a spring is modified according to the activation of outputs in the two cells attached to the spring and equal to $L = (1 + A_{\text{max}}(s_1 + s_2))L_0$, where A_{max} is a global parameter representing the maximum amplitude of change from one cell ($A_{\text{max}} = 0.15$) and s_1 and s_2 are corresponding activations of each cell's actuation output. Just as during the developmental stage, for performance reasons, the frequency of updates of networks in cells continues to be an order of magnitude lower than that of the underlying physics. To avoid jerky movements that would result from potentially sudden changes in the resting lengths of springs after each network update, we modify the resting lengths progressively, in between subsequent network updates.

Genetic encoding and genetic algorithm

To encode and evolve networks controlling animats' cells we employed NEAT (neuroevolution of augmenting topologies Stanley and Miikkulainen, 2002), which can be considered one of the state of the art methods of evolving artificial neural networks. In the NEAT method, genomes are represented as a list of nodes with their associated type (in-

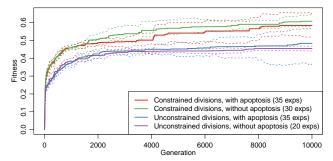


Figure 2: Comparing evolvability of target morphologies while using two different modes of cellular division with cellular death (apoptosis) either enabled or disabled. Solid curves show medians of best individuals in series of experiments of each type, dashed curves show bootstrapped 95% confidence intervals.

put, output, hidden), activation function (we allowed for sigmoidal, gaussian, step, linear, sine and abs functions) and a list of connections. The NEAT algorithm keeps track of innovations history and uses it to perform crossover between genomes. It also uses fitness sharing approach with the goal of preserving diversity and protecting new solutions before they have to compete with the rest of the population. We used population sizes of 300 and runs of 2000 generations (body-controller coevolution) or 10 000 generations (evolution of desired shapes). Initial population was created as a fully connected feed-forward network with a hidden layer and random weights.

Results

To verify the ability of the proposed approach to evolve structures made of hundreds of cells, we chose to first attempt evolving embryos that would be required to develop into a simple shape (such as a disc or a cross), using a fitness function that would reward similarity of the obtained morphology to the target shape. This allowed us to gain initial insight into how evolutionary process shapes development and to adjust physical constants so that development proceeds in a stable manner. Only then we have attempted coevolving morphologies and controllers for locomoting animats.

Evolving desired morphologies

To evolve embryos with desired shapes, we discretized target shapes into pixels and used a straightforward fitness function defined as $f=\frac{i-o}{c}$, where c was the count of pixels that a shape consisted of, i the count of the shape's pixels that were covered by the embryo at the end of developmental process and o was the count of pixels occupied by cells outside the target shape.

In our first attempt, similarly to the previous works (Joachimczak and Wróbel, 2012), we allowed cells to simply divide whenever activation of the output associated with

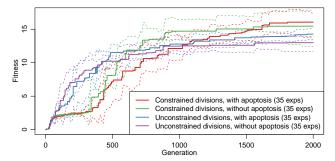


Figure 3: Comparing evolvability of animat gaits while using two different modes of cellular division with cellular death (apoptosis) either enabled or disabled. Solid curves show medians of best individuals in series of experiments of each type, dashed curves show bootstrapped 95% confidence intervals.

division crossed a threshold. Although we were able to obtain highly fit individuals for simple target shapes such as an ellipsoid or a cross, analysis of their developmental process revealed that it was rather simplistic: cells would tend to divide in synchrony at the maximum allowed frequency, with their orientation vectors hardly showing any organization or relevance to the target shape. Rather than a product of organized growth towards a desired shape, a good match of morphology to the target shape was found to be simply the result of apoptosis trimming excess cells after growth (Fig. 1a). As the developmental process in which there is little or no differentiation between cellular behavior did not seem as a good starting point for evolving complex animats, we pursued other approaches that could facilitate cells taking control over timing and directionality of their divisions.

Ultimately, we have introduced a simple modification to the division process: a cell would not divide in the direction that was already occupied by another cell (it could however divide if there was a small amount of space available, the physics would push obstructing cells to the sides). Assembling an embryo of a shape other than line of cells would now require active control over the direction and timing of cell division, thus enforcing emergence of differentiation in cellular behavior already at the very beginning of evolution. Another useful property of this mode of division is that due to the fact that nearby cells receive largely similar external signals, cells tend to grow off the main body in groups sharing similar orientation vectors, facilitating creation of animat's protrusions. Shapes obtained in this manner had on average similar quality compared to the unconstrained approach (when using cross target shape, of the 20 runs, 10 would produce a 4-armed shape, while remaining ones would result in 3- or 2-armed shapes), yet the developmental process would rely on a clearly more organized, progressive growth (Fig. 1b).

To quantitatively investigate the difference between the two approaches, we compared quality of solutions obtained in repeated evolutionary runs using both methods. Surprisingly, the most fit individuals of all the runs were obtained using unconstrained divisions, although the median quality of these solutions was found to be lower (Fig. 2, red and blue curves). Since we hypothesized that the main reason unconstrained divisions can produce high fitness solutions is the ability to selectively remove cells at the end of development, we have performed the same comparison rerunning evolution with apoptosis disabled. High quality solutions with unconstrained development became entirely absent, and the medians of fitness distributions of best individuals at the end of runs were clearly different and in favor of constrained development ($p < 3.01^{-9}$, Wilcoxon two-tailed rank test; green and violet curves in Fig. 2). We interpret it as a confirmation that allowing for unconstrained divisions and apoptosis makes evolution focus its search on high quality yet trivial and unlikely evolvable solutions, in which the activation of the output responsible for apoptosis is simply a direct function of cell's spatial location.

To see however if this observation carries from the evolution of directly specified morphologies to the evolution of soft-bodied animats, we have later performed an analogous set of experiments on the latter problem. The results do exhibit similar pattern (Fig. 3). Although when apoptosis was enabled, the observed higher median quality of solutions obtained in experiments using constrained division was not confirmed to be significantly different from unconstrained division (p = 0.09), the difference was significant when apoptosis was disabled ($p < 9.63^{-4}$). Furthermore, visual analysis of individuals obtained with unconstrained cellular divisions also suggests that even with apoptosis enabled, their shapes would tend to be simpler and more convex. This also explains why this mode of division produces higher fitness individuals during the first few hundreds generations: before the constrained developmental process evolves enough control over cellular divisions to produce useful morphologies, unconstrained development has already discovered many largely circular morphologies that can move. Yet, this bias for convexity makes it harder to discover more complex solutions later on.

All animats in the experiments discussed later in the paper were therefore evolved using constrained cellular divisions.

Evolving soft-bodied animats

We evolved individuals for their ability to produce gaits in a 2 dimensional environment with gravity, with fitness function corresponding to the distance traveled during the length of the locomotion stage. Evaluation would proceed by first developing an embryo during developmental stage, converting it to a soft-bodied animat and simulating its behavior during locomotion stage. Coevolving controllers with unconstrained morphologies posed however a set of additional challenges that needed to be addressed, so that the genetic algorithm would avoid certain local minima.



Figure 4: Examples of two developmental processes leading to creation of the animats seen in Fig. 5ab. The last frame of each sequence shows morphology after the embryo is converted into a soft-bodied animat. Figure videos: http://youtu.be/Q6XXohgPP4A, http://youtu.be/1c-V-ttMIZI

Fitness function In particular, the most straightforward approach of calculating displacement of individual's center of mass during locomotion stage was found to not work well and to be very deceptive for evolutionary search. The prime cause of that was the fact that physical environment of the locomotion phase differs from the developmental phase by the presence of gravity. Enabling gravity at the beginning of simulation thus causes animat to first compress under its own weight. If displacement of the center of mass was used as a fitness function, evolution was found to quickly exploit this by focusing evolutionary search on vertically elongated structures that collapse and produce horizontal displacement by bouncing of the ground in often spectacular, yet not particularly useful ways: clearly, it is easier to discover genomes that produce elongated, vertical morphologies than individuals that can coordinate their actuation well enough to produce some movement. An obvious solution of giving an animat a certain fixed time to equilibrate did not entirely solve the problem either, as the genetic algorithm turned out to be very successful in discovering vertical structures that can balance themselves for even long periods of time and start collapsing exactly after this fixed period of time. In the end, we addressed this problem twofold. Firstly, we enabled drag, proportional to the speed squared, to act on the nodes during equilibrating. This largely eliminated exploitation of physics with tall animats that reach very high speeds and forces when collapsing under gravity. Secondly, we would start actual evaluation of the motion only after the center of mass was determined to be immovable (within a precision of a set threshold) for a certain number of steps.

Guiding the exploration of the search space towards individuals capable of any locomotion was yet another problem that required special attention. As we wanted to avoid introducing bias on the search by using handcrafted seed individual, we attempted to start evolutionary process with simple randomized genomes. A random network is not however guaranteed to make the first cell divide at all, and even if it does so, the resulting morphology is not likely to display any actuation. This essentially means that an individual that does not even start divisions and a potentially promising morphology with pronounced appendages, but incapable of

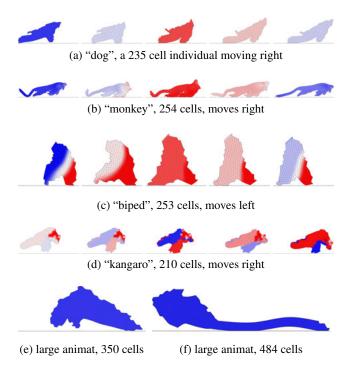


Figure 5: Motion snapshots of example obtained soft bodied animats. Colors show actuation: red - spring resting lengths are increased, blue - decreased, white - they are at their default, initial lengths. Figure videos: http://youtu.be/7y8HixKBO8g

actuation, would both have the same zero fitness assigned. To help search algorithm differentiate between these two cases, we have introduced these simple measures: genomes that would produce a valid multicellular embryo that terminates its development would receive a base fitness of 0.1. On top of that, a small reward (up to 0.1) would be added if the outputs controlling the resting length of springs would change their value during evaluation, proportionally to the amount of actuation. Only then, the reward corresponding to the actual displacement of animat's center of mass would be added on top of these basic rewards. This allowed to focus evolutionary search on individuals potentially capable of movement early on.

Evolved animats Having addressed the above issues, we found the system to produce surprisingly rich and efficient morphologies with relative ease, with almost all runs resulting in animats exhibiting gaits. A few examples of more interesting individuals are shown in Fig. 5 and the quality of their gaits can perhaps be best judged through the supplementary videos linked from the figure. Observed morphologies include clearly "2-legged" creatures (Fig. 5abc) as well as "1-legged" jumpers (Fig. 5d) or creatures with a "tail" (Fig. 5bf). Occasionally, circular, rolling individuals were also observed (albeit not shown here). Most importantly, it was common for the animats to evolve some forms of appendages to support their body and, in difference from

our previous work, appendages were not limited to a single cell or a few cells, but were clearly pronounced structures growing out of the main body and sometimes consisting of dozens of cells. Developmental process was also observed to be much more sophisticated than in our previous work, with differential growth rates in different body regions and occasional use of apoptosis (two examples of self assembling embryos are shown in Fig. 4). Patterns of cellular actuation would also often be differentiated spatially along the body, making cells contract and expand in a non trivially organized manner (e.g., Fig. 5d).

The animats shown in Fig. 5abcd were evolved with a 256 cells limit and were found to grow up to the higher end of allowed sizes. We expected this, given there was no cost of actuation and the fitness was an absolute measure of distance (i.e., not in relation to the body size), putting small individuals at disadvantage. We have also attempted evolving animats with a 1024 cell limit and have observed some larger solutions (Fig. 5ef), though the animats would often struggle to actuate under their increased weight and decrease in size as the evolution progressed, an issue we think is not so much a limitation of the system's scalability but rather requires readjusting physical constants that control the elasticities of springs and the amplitude of actuation.

Optimizing for material use Energy efficient gaits of biological animals are not only the result of work performed by actuators (muscles), but also body's ability to effectively convert kinetic energy into potential and elastic energy and vice versa, leading to motion patterns strikingly different from movements typically associated with robots, where movement is driven by actuators setting, e.g., precise angles of every joint. Creating animats out of elastic, compressible materials offers rich opportunities to invent efficient motion patterns that rely on converting kinetic energy into elastic energy and vice versa (as also recently explored in Cheney et al., 2013). As a trivial modification of the system we allowed one of the outputs to specify whether a cell is passive (activation < 0) or capable of actuation (activation >= 0). We have then modified the fitness function, so that it promotes reduced number of actuating cells. As a reward for reducing the use of actuators, the fitness component previously calculated directly based on the displacement of the center of mass, could now be increased up to twofold (though the maximum reward would only happen if a single cell was used for actuation). From the single preliminary experiment we performed, 5 of the 10 evolutionary runs resulted in animats that reduced their use of actuating material and relied on the elasticity of the rest of the body to produce motion patterns. Examples of their morphologies are shown in Fig. 6.

Evolution of developmentally plastic animats One of the main reasons behind the creation of a system that explicitly simulates multicellular development to evolve ani-

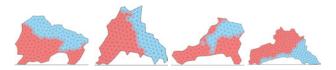


Figure 6: Four individuals obtained using fitness function rewarding both the distance and reduced use of active material. Red: cells capable of actuation, blue: passive elastic material. Videos of individuals in motion: http://youtu.be/VcClTGlTWkg

mat morphologies is to further our understanding of the relationship between development and evolution, by performing relevant evolutionary experiments *in silico*. In particular, we are interested in investigating the plasticity of animal development, both on the evolutionary level and of a single individual. The latter phenomenon, developmental plasticity, is understood as the capability of developmental process to take a different course and produce different phenotypes, each fine-tuned to a different state of environment they expect to find themselves (known as morphs, see e.g., Gilbert and Epel, 2008 for examples).

As a preliminary evaluation of the capability of the presented system to produce developmentally plastic animats, we have introduced a new type of external signal and a second type of environment, in which horizontal surface is replaced with a slope. Each individual was then evaluated twice, once in each of the environments. Whether an individual would be evaluated on a sloped or on a horizontal surface was then signaled by a value of 1 or 0 of the external signal, from the beginning of the developmental process. The fitness of every individual was then calculated as a square root of the product of fitnesses obtained in the two scenarios. The component fitness was calculated as in previously described experiments, with the exception that only animats moving right would receive a non zero fitness (as otherwise they would be rewarded for rolling down the slope). Although we were able to obtain some developmentally plastic animats which would produce 2 different phenotypes for each of the two environments, starting the evolutionary process became much more difficult, as finding individuals capable of moving up the slope even slightly without having previously invented some gait, was unlikely. Most of individuals in the early generations, even if capable of some minimal displacement on a flat surface, on a slope would simply collapse to a side, resulting in no displacement reward assigned and effectively random exploration of the genome space. Better results were obtained when we modified the fitness function so that individuals were first evolved to produce a gait on a horizontal surface, and then, since generation 500 until generation 2000, we progressively increased the angle of the slope until its target value. Fig. 7 shows an example of 2 individuals (their two morphs) obtained in this way.

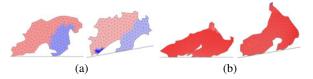


Figure 7: Two examples of developmentally plastic animats. Each animat in the pair is encoded by the same genome, which produces a different phenotype for a horizontal and sloped environment. Color indicates deviation of spring resting length at the time snapshots were taken. Figure videos, including development: http://youtu.be/lgbe2QbU0as

Conclusions and future work

In this work we have investigated the feasibility of applying fine-grained multicellular development to coevolve morphologies and controllers of virtual robots whose bodies consist of hundreds rather than dozens of independent components, i.e. cells. We did so by attempting to scale up the ideas introduced in our prior work, which however required us to reapproach the problem to facilitate growth of more complex structures. Ultimately, this allowed us to find out that a relatively simple developmental system employing neural networks as cell controllers, coupled with an improved approach to cellular division and simulated softbodied physics is all that is needed to enable evolution of much more complex and diverse forms.

We see the richness of morphologies and gaits that emerged in our experiments as an example of still largely untapped potential of multicellular development in the domain of morphology-controller coevolution. Naturally, when it comes to simply designing a robot, employing higher level abstractions of development, where components can be defined by geometric primitives instead of having to be assembled through a coordinated action of dozens of cells, would outperform our approach when it comes to computational complexity, especially if 3 dimensions were to be considered. Nonetheless, since this is not how the most evolvable system known so far, i.e., biological development works, we are willing to argue for the importance of pursuing evolvable cell-level abstractions of development, as a way to advance our understanding of development and its evolution. We thus plan to investigate further what aspects of the system facilitate evolvability and, in particular, lead to emergence of genotype-phenotype mappings in which relatively small changes in the genomes can lead to organized high level changes, such as enlargement or cloning of a whole body structure (see Doursat, 2009 for an example of such a system).

Finally, having shown the capability of the system to evolve developmentally plastic animats, we plan to investigate this aspect of development further, by analyzing how a single genotype-phenotype mapping diverges into two or more phenotypes (morphs) (see e.g., Palmer, 2012, for a recent discussion of the postulated hypotheses). Another form

of developmental plasticity that can be explored with this system is the capability of producing metamorphic individuals which, when triggered, can convert one phenotype into another, a phenomenon displayed by a wide variety of organisms that undergo staged life cycles in different environments (e.g., amphibians). From an engineering point of view, such artificial systems could be seen as a variant of evolutionary design, where instead of evolving single solution to the posed problem, genomes are evolved to encode a group or a continuum of solutions, each best adapted to a different variation of a given problem.

Acknowledgments

This work was supported by the Japan Society for the Promotion of Science (JSPS) through the JSPS Fellowship for Foreign Researchers, the JSPS Grant-in-Aid for Scientific Research. High performance computing resources were provided by the Interdisciplinary Center for Molecular and Mathematical Modeling (ICM, University of Warsaw; project G33-8) and the Tri-city Academic Computer Center (TASK).

References

- Auerbach, J. E. and Bongard, J. C. (2012). On the relationship between environmental and mechanical complexity in evolved robots. In *Artificial Life 13*, pages 309–316. MIT Press.
- Bongard, J. C. and Pfeifer, R. (2003). Evolving complete agents using artificial ontogeny. In Hara, F. and Pfeifer, R., editors, *Morpho-functional Machines: The New Species*, pages 237–258. Springer Japan.
- Carroll, S., Grenier, J., and Weatherbee, S. (2004). From DNA to Diversity: Molecular Genetics and the Evolution of Animal Design. Wiley-Blackwell, second edition.
- Cheney, N., MacCurdy, R., Clune, J., and Lipson, H. (2013). Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO)* 2013.
- Cussat-Blanc, S., Sanchez, S., and Duthen, Y. (2012). Controlling cooperative and conflicting continuous actions with a gene regulatory network. In *Computational Intelligence and Games* (CIG), 2012 IEEE Conference on, pages 187–194. IEEE.
- Doursat, R. (2009). Facilitating evolutionary innovation by developmental modularity and variability. In *Proc. of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 683–690. ACM.
- Doursat, R., Sayama, H., and Michel, O. (2013). A review of morphogenetic engineering. 12(4):517–535.
- Gilbert, S. F. and Epel, D. (2008). *Ecological Developmental Biology*. Sinauer Associates, first edition.
- Hiller, J. D. and Lipson, H. (2010). Evolving amorphous robots. In *Artificial Life XII: Proc. of the 12th International Conference on the Simulation and Synthesis of Living Systems*, pages 717–724. MIT Press.
- Joachimczak, M., Kowaliw, T., Doursat, R., and Wróbel, B. (2013). Evolutionary design of soft-bodied animats with decentralized control. *Artificial Life and Robotics*, 18(3-4):152–160.

- Joachimczak, M. and Wróbel, B. (2010). Processing signals with evolving artificial gene regulatory networks. In *Artificial Life XII: Proc. of the 12th International Conference on the Simulation and Synthesis of Living Systems*, pages 203–210. MIT Press
- Joachimczak, M. and Wróbel, B. (2012). Co-evolution of morphology and control of soft-bodied multicellular animats. In *Proc.* of the 14th International Conference on Genetic and Evolutionary Computation, GECCO '12, pages 561–568. ACM.
- Komosinski, M. and Ulatowski, S. (1999). Framsticks: towards a simulation of a nature-like world, creatures and evolution. In *Proc. of Fifth European Conference on Artificial Life (ECAL 1999)*, volume 1674 of *LNAI*, pages 261–265. Springer-Verlag.
- Kowaliw, T., Grogono, P., and Kharma, N. (2004). Bluenome: A novel developmental model of artificial morphogenesis. In *Genetic and Evolutionary Computation GECCO '04*, pages 93–104. Springer Berlin / Heidelberg.
- Lipson, H. and Pollack, J. B. (2000). Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974–978.
- Meng, Y., Zhang, Y., and Jin, Y. (2011). Autonomous self-reconfiguration of modular robots by evolving a hierarchical mechanochemical model. *IEEE Comput. Intell. Mag.*, 6(1):43–54.
- Nicolau, M., Schoenauer, M., and Banzhaf, W. (2010). Evolving genes to balance a pole. In *EuroGP: 13th European Conference* on *Genetic Programming*, volume 6021 of *LNCS*, pages 196– 207. Springer.
- Palmer, A. R. (2012). Developmental plasticity and the origin of novel forms: Unveiling cryptic genetic variation via use and disuse. J. Exp. Zool., 318(6):466–479.
- Pilat, M. L., Ito, T., Suzuki, R., and Arita, T. (2012). Evolution of virtual creature foraging in a physical environment. In Artificial Life XIII: Proc. of the 13th International Conference on the Simulation and Synthesis of Living Systems, pages 423–430. MIT Press.
- Roggen, D. and Federici, D. (2004). Multi-cellular development: Is there scalability and robustness to gain? In *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *LNCS*, pages 391–400. Springer Berlin Heidelberg.
- Schramm, L., Jin, Y., and Sendhoff, B. (2011). Emerged coupling of motor control and morphological development in evolution of multi-cellular animats. In *Advances in Artificial Life. Darwin Meets von Neumann: Proc. of the 10th European Conference on Artificial Life (ECAL 2009)*, volume 5777 of *LNCS*, pages 27–34. Springer.
- Shepherd, R. F., Ilievski, F., Choi, W., Morin, S. A., Stokes, A. A., Mazzeo, A. D., Chen, X., Wang, M., and Whitesides, G. M. (2011). Multigait soft robot. *Proc. Natl. Acad. Sci. U. S. A.*, 108(51):20400–20403.
- Sims, K. (1994). Evolving virtual creatures. In *Proc. of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 15–22. ACM Press.
- Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evol. Comput.*, 10(2):99–127.
- Stanley, K. O. and Miikkulainen, R. (2003). A taxonomy for artificial embryogeny. *Artif. Life*, 9(2):93–130.

Adapting Morphology to Multiple Tasks in Evolved Virtual Creatures

Dan Lessin, Don Fussell, and Risto Miikkulainen

The University of Texas at Austin, Austin, TX 78712 dlessin@cs.utexas.edu

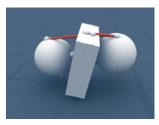
Abstract

The ESP method for evolving virtual creatures (Lessin et al., 2013) consisted of an encapsulation mechanism to preserve learned skills, a human-designed syllabus to build higherlevel skills by combining lower-level skills systematically, and a pandemonium mechanism to resolve conflicts between encapsulated skills in a single creature's brain. Previous work with ESP showed that it is possible to evolve much more complex behavior than before, even when fundamental morphology (i.e., skeletal segments and joints) was evolved only for the first skill. This paper introduces a more general form of ESP in which full morphological development can continue beyond the first skill, allowing creatures to adapt their morphology to multiple tasks. This extension increases the variety and quality of evolved creature results significantly, while maintaining the original ESP system's ability to incrementally develop complex behaviors from a sequence of simpler learning tasks. In the future, this method should make it possible to build EVCs with complex and believable behavior.

Introduction

Since their introduction two decades ago by Sims (1994), evolved virtual creatures (or EVCs; Figure 1) have had a significant impact on multiple fields, including graphics, evolutionary computation, artificial life, and robotics (Shim and Kim, 2003; Lehman and Stanley, 2011; Miconi, 2008; Lipson and Pollack, 2000). But despite their many incarnations in the intervening years, their behaviors have not grown more complex. Sims' original work demonstrated light following. This level of complexity has been approximately matched multiple times since (Miconi, 2008; Pilat and Jacob, 2010), but was never clearly exceeded until the ESP system of Lessin et al. (2013). ESP was shown to construct EVC behaviors approximately twice as complex as any seen before, measured as the number of discriminable behaviors in the creature's repertoire. Moreover, in principle there is no upper limit on behavioral complexity yet established within this new framework.

The ESP system is named for its three principal components-encapsulation, syllabus, and pandemonium-defined as follows:



(a) Initial locomoting creature.

(b) Heavy smashing arms.

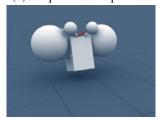




(c) Smashing flail arms.

(d) Jump with anti-tip limbs.





(e) Smashing tail, stabilizers.

(f) Jump with heavier body.

Figure 1: Adapting EVC morphology to multiple tasks. (a) A creature adapted for locomotion. From this creature, creatures (b) through (f) were evolved using the extended ESP method described in this paper. Each of them has developed a new technique (with corresponding morphological changes) for accomplishing an additional task—in this case, delivering a strike to the ground—while still maintaining the ability to perform the initial skill (locomotion) to prescribed levels. In the original ESP system, these secondary adaptations would have been impossible.

Once a new skill is learned, it can be encapsulated, which
preserves that ability and makes it easily accessible to future evolution.

- A human-designed syllabus is used to direct the sequential acquisition of these component skills toward the larger goal of achieving new, more complex behaviors, building up hierarchically from simpler ones.
- A *pandemonium*-like mechanism (Selfridge, 1958) is employed to resolve conflicts between competing encapsulated skills in the increasingly complex brain.

The initial ESP implementation did achieve its goal of breaking the behavioral-complexity barrier for EVCs. However, it applied only to a significantly restricted case—one in which some of the most important morphological changes (those to the creature's skeletal segments and joints) were prohibited after the first skill's evolution was complete. (Note, however, that muscle drives and photoreceptors—described below—could continue to be evolved throughout all evolutionary stages, even in the original system, since they can be added without disrupting existing abilities.)

This is a serious limitation. For example, what if a creature is evolved for an initial skill such as locomotion, then is asked to adapt to a largely orthogonal skill such as reaching up to a high target? That creature may or may not have the required morphological capacity for performing the second task, as determined only by the accidents of evolution.

This paper introduces a significantly extended version of ESP, in which a retesting and reconciliation scheme replaces previous absolute limitations on morphological evolution. Morphology is thus fully evolved to suit the requirements of more than just a single skill.

In the following sections, this new ESP implementation is described. The results demonstrate a significant increase in the useful variety and quality of evolved creatures, while the ability to incrementally develop complex behaviors from a sequence of simpler learning tasks is maintained.

The Underlying EVC Implementation

Both the ESP implementation of Lessin et al. (2013)—called *original ESP* in this paper—and its improvement that is presented in later sections—called *extended ESP*—were built on a reimplementation of Sims' original work (1994) that will be referred to as *the underlying EVC implementation*. That underlying EVC system—along with some important novel extensions—is described in the balance of this section, and a representative sample of its results is shown in Figure 2.

Evolutionary Algorithm

A conventional evolutionary algorithm is used, with elitism, fitness-proportionate selection, and rank selection (Mitchell, 1998). In addition, the most challenging tasks employ shaping (Urzelai et al., 1998; Gomez and Miikkulainen, 1997). Fitness is evaluated in a physically simulated virtual environment implemented with NVIDIA PhysX.

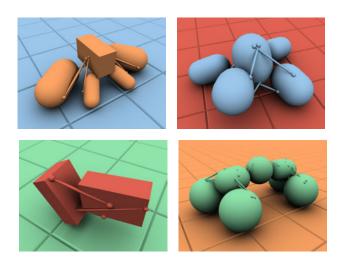


Figure 2: Typical results from the underlying EVC system. These examples were all evolved to complete a forward locomotion task—a common baseline result for EVCs.

Morphology

As in Sims' original work, creature morphology is described by a graph-based genotype, with graph nodes representing body segments, and graph edges representing joints between segments By starting at the root and traversing the graph's edges, the phenotype is expressed. Reflexive edges as well as multiple edges between the same node pair are allowed, making it possible to define recursive and repeated body substructures easily. In addition, as in Sims' work, reflection of body parts as well as body symmetry are made easily accessible to evolution, with only a single mutation required to produce each of them. In this implementation, all PhysX primitives are available for use as body segments: boxes, spheres, and capsules. Joints between segments may be of most of the types offered by PhysX, specifically: fixed, revolute, spherical, prismatic, and cylindrical. In contrast to the typical technique of separately evolving explicit joint limits, most limitations on joint movement are provided implicitly by creature structure through natural collisions between adjacent segments.

In addition to the typical segments and joints, the implementation of the underlying EVC system also evolves muscle drives and photoreceptors, as described below.

Muscles

In a break with traditional evolved virtual creature systems, which typically use forces exerted directly at joints, the underlying EVC system of this paper uses simulated muscles as actuators. Each muscle ((b) in Figure 3) is defined by two attachment points on adjacent segments, along with a maximum strength value. In simulation, the muscle is implemented as a spring, with muscle activation modifying the spring constant. In addition to acting as an effector, each muscle also produces a proprioceptive feedback signal based

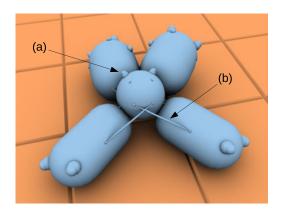


Figure 3: Photoreceptors (a) and muscles (b) bring sensing and actuation to creatures in the underlying EVC system. For both, function depends upon placement, so creature form develops meaningfully as capabilities are evolved.

on its current length. For each muscle, two nodes are added to the brain: one that accepts an input to set the muscle's activation, and another that makes the muscle's proprioceptive output signal available to the rest of the brain. Muscle drives benefit EVCs in several ways: they can be used even on creatures without joints; they only need to exist where they are useful, not at every degree of freedom of every joint; and they have the potential to embody some degree of control intelligence, with benefits for both aesthetics and the reduction of cognitive load (Lessin et al., 2014).

Photoreceptors

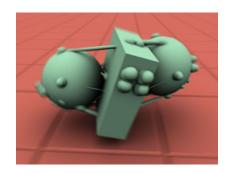
For tasks involving light sensing, creatures are allowed to develop simple photoreceptors ((a) in Figure 3), defined only by a direction from the center of their parent segment. This direction indicates a location on the creature's surface as well as an orientation for the receptor. The signal produced by the receptor is determined by light strength, distance, occlusion, and the difference between the direction to the light and the sensor's orientation. Multiple lights are allowed. For each photoreceptor in the body, a corresponding brain node is added which makes the receptor's output signal available to the rest of the brain.

Control

In a manner which is again very similar to that of Sims, creature control is provided by a brain composed of a set of nodes connected by wires (as in Figure 5a). Nodes receive varying numbers of input wires, and use their inputs to compute an output value (always in the range [0,1]) which may be sent to other wires. Signals originate from sensors in the body as well as certain types of internal brain nodes, travel through the network of internal nodes and wires, and ultimately control the operation of actuators (muscles) in the physically simulated body. For each step of physical simulation, control signals move one step through the brain.

In addition to the special node types for muscles and photoreceptors (described above) and for encapsulation (described in the *Encapsulation* subsection), the following node types are allowed: sinusoidal, complement, constant, scale, multiply, divide, sum, difference, derivative, threshold, switch, delay, and absolute difference.

The Original ESP System



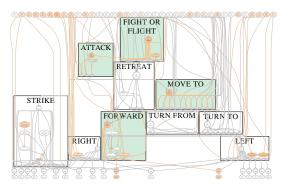


Figure 4: The body and brain of a creature evolved using the original ESP system. This creature achieved a level of behavioral complexity approximately double the state of the art. In addition to following a light, it was able to attack it or flee from it, as part of a hierarchical fight-or-flight behavior.

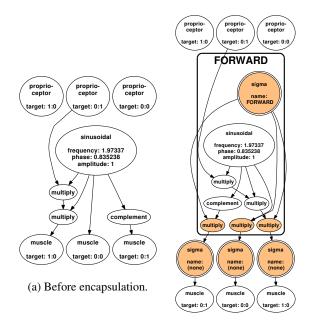
The original ESP system (Lessin et al., 2013) consists of three elements added to the underlying EVC system: encapsulation, syllabus, and pandemonium. In this section, each of these components is described in detail.¹

Encapsulation

The first important element of the ESP system is a mechanism to *encapsulate* newly learned skills (Figure 5). This element accomplishes two important goals: It ensures that previously learned skills (and the body components on which they rely) are preserved, and it makes these skills easily accessible to future evolutionary development.

Figure 5a depicts a brain evolved for forward locomotion, and Figure 5b shows the result of encapsulating it. First, the nodes that compute the old skill have been preserved and

¹For a video overview, see: http://youtu.be/dRLNnJlT8rY



(b) After encapsulation (with new nodes shaded).

Figure 5: Encapsulation. The automated encapsulation of an evolved skill—in this case, forward locomotion—ensures that it will persist throughout future evolution, while also allowing it to be easily activated as a unit by future skills.

locked (meaning that they have been marked to disallow any changes by future evolution). Second, a new *multiply* node has been inserted into every output wire leaving the encapsulated skill. The internals of the skill will continue to function as before, always trying to perform their forward locomotion task, but now, a second signal sent to each new multiply node will modify those outgoing forward-locomotion control signals, scaling them by a number in [0,1]. Third, a single controlling node (called a *sigma node* for its function as a summation of zero or more inputs) is added, which sends its output to all of the new multiply nodes. So for each signal s_i leaving a node in the FORWARD LOCOMOTION skill (such as the *complement* node), the new signal after encapsulation (s_i') is computed as $s_i' = \sigma s_i$ where σ is the output of the controlling sigma node.

Now, with encapsulation complete, the entire forward locomotion skill can be activated and deactivated as a unit by using the controlling sigma node just as if it were a single muscle in the creature's body. (Incidentally, note that this brain's actual muscle nodes have been hidden behind additional sigma nodes to allow future evolution to share control over them when appropriate.)

With the ability to preserve completed skills for easy future access, the next issue is how they may be acquired in sequence to achieve a larger learning objective.

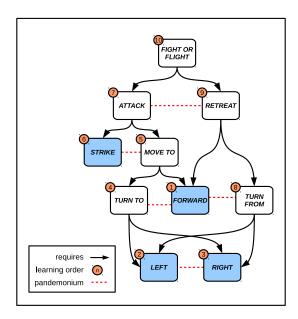


Figure 6: An example syllabus as a graph. Graph nodes represent individual subskills to be learned, directed edges indicate dependency between subskills, and the numbering indicates a proposed learning order that satisfies the dependency requirements. Pandemonium relationships are indicated by dashed red lines. The shaded nodes (called *leaf nodes*) affect only the body, rather than other nodes, and constitute the focus of the extended ESP system discussed in this paper.

Syllabus

While it is certainly possible for human students to learn a complicated topic independently, even advanced learners would be expected to develop further and with greater ease with the benefit of an expert-designed syllabus. The syllabus acts as a sequence of landmarks through the space of possible solutions, decomposing the larger learning task into a succession of more manageable steps between these way-points.

In the ESP system, the *syllabus* consists of an ordered sequence of fitness goals used to reach the ultimate, larger goal. This collection of intermediate goals (each one defined by a fitness function) is designed by a human expert with the aim of making attainable goals more reliably learnable, and bringing previously unattained goals within reach.

For example, assume that you want to evolve a virtual creature with some of the behavioral complexity demonstrated in an internet cat video.² Rather than simply drifting smoothly toward a target, this creature might run to the target, then strike it, and perhaps even run away if the target is perceived as threatening. Without a syllabus, a single fitness test evaluating such a complex collection of skills might be constructed, but evolutionary progress would be unlikely.

²E.g.: "THE BEST CAT VIDEO YOU'LL EVER SEE" [sic], http://youtu.be/20mrEtabOLM

Consider, instead, how this complex behavioral goal could be broken down into an ordered sequence of smaller learning tasks. The clearly achievable goal of locomotion will be the first target. The ability to turn left and the ability to turn right are of a similarly manageable difficulty, and will be attempted next. Then, with left and right turns mastered, and the ability to develop photoreceptors, it is relatively straightforward to maintain orientation toward a light source. And with the ability to face a light and the ability to move forward, navigating to that light might be a similarly achievable goal. Proceeding in this manner, a knowledgeable human designer might produce a sequence of subskills to be learned. Each subskill would be attainable with basic EVC methods, and earlier subskills would serve to make later skills easier to learn.

This information may be conveniently summarized in a graph, encompassing subskills to be learned, dependency between subskills, learning order, and pandemonium, as seen in Figure 6. In fact, the syllabus in this figure was used by Lessin et al. (2013) to produce the creature illustrated in Figure 4. This creature achieved a level of behavioral complexity approximately double the state of the art for evolved virtual creatures at that time. More specifically, not only was it able to move to a light target, as previous creatures had done, but also strike the target upon reaching it, and flee from the target when it became dangerous, doubling the complexity of behaviors in prior work.

At this point, a complex skill can be broken into smaller subskills, and those subskills can be cumulatively acquired, but a potential problem still remains: How are competing signals from the multiple sub-brains within a single creature resolved?

Pandemonium

Consider the following example based on the syllabus graph of Figure 6. A creature evolved through this syllabus will ultimately have parts of its brain devoted to both left and right turns. But it seems unlikely that both of these abilities should ever be used at the same time. So the syllabus designer might place the left and right-turn skills in a *pandemonium* relationship with each other (Selfridge, 1958), meaning that whichever one is most active at any given moment will be allowed to send its output at full strength, and the other will have its output entirely suppressed. Under a system like this, sub-brains within the creature can compete for overall control, with little risk of sabotaging the usefulness of the entire brain's function. In Figure 6, a full set of pandemonium relationships is indicated by red dashed lines between subskill nodes.

Although this original ESP system achieved more complex behavior than before, body segments and joints could not continue to adapt beyond the first skill's completion. In the next section, a more general form of the ESP system is described that makes this possible.

The Extended ESP System: Adapting Morphology to Multiple Tasks

In this section, a new version of ESP is described with extended evolution of morphology.

Replacing Morphological Constraints with Retesting

The initial implementation of the ESP system enforced strict limits on morphological changes after the completion of the first skill: Although changes to muscles and photoreceptors were allowed, segments and joints were fixed. Due to this constraint, previously learned skills could be expected to work reliably throughout the syllabus-based construction. On the other hand, this limitation makes it difficult to develop certain abilities later. For example, a creature may succeed in developing forward locomotion and the ability to turn left, but—due to the construction of a certain joint evolved for locomotion—be unable to learn to turn right, even after many generations of evolution.

Luckily, this limitation was undertaken only to make an initial success in the original system easier to achieve. It can be removed simply by expanding and modifying the fitness evaluations applied during learning: Instead of locking segments and joints after the first skill is developed, successive skills could be allowed to change these attributes, as long as new testing shows that such changes will not invalidate earlier abilities.

However, such an increase in testing threatens to make an already computationally demanding problem significantly more difficult, especially because the system is intended to be open ended. Assuming n skills and one independent test for each skill, full retesting of all previous skills at each step of the syllabus would produce an $\mathcal{O}(n^2)$ growth in the required testing, instead of the current linear growth.

Fortunately, the retesting can be considerably reduced by focusing it where it matters. Consider again the syllabus graph shown in Figure 6. The skills that have a direct influence on the creature's body are shaded, and will be referred to as *leaf* skills. These are: FORWARD LOCOMOTION, LEFT TURN, RIGHT TURN, and STRIKE. Once these skills are successfully established, the remaining non-leaf skills can be evolved independently (in an order that meets dependency requirements), without the need for any retesting. This approach stops the $\mathcal{O}(n^2)$ growth in testing requirements significantly earlier than would otherwise be possible—in the syllabus of Figure 6, for example, after four skills instead of 10 (assuming all leaf skills are learned first).

New ESP Algorithm

This subsection describes the implementation of the new, more general form of the ESP algorithm, taking advantage of the leaf skills. The method is comprised of two stages. The first stage consists of a fixed number of generations during which the new skill's control and body evolves, as de-

scribed in Algorithm 1. During this stage, existing encapsulated skills in the brain do not change, but if any morphological changes reduce these skills' fitness beyond a preset limit, the creature will be marked as unfit. In this way, the new skill is given free rein to adapt the body to its needs, provided that sufficient ability in all existing skills is retained.

The second stage runs for a fixed number of generations for each of the old skills, during which the morphology is temporarily locked–ensuring that the abilities achieved by the new primary skill are preserved–and each of the already existing skills gets a chance to reconcile itself to the new body (Algorithm 2). Since the morphology is fixed, these skills can develop completely independently–each skill can adapt to the new body, without degrading any of the other skills in the brain.

Proceeding in this manner, this extension of the ESP algorithm allows new leaf skills to seek their own adaptations to morphology as well as control, with a reasonable expectation that—as in the old system—existing skills will be maintained, allowing abilities to accumulate incrementally as in the original ESP.

```
Algorithm 1: Full evolution of morphology and control
 for new skill s'
1 foreach generation do
      foreach individual in the population do
2
3
          mutate morphology;
4
          mutate control for new skill s';
5
          foreach existing skill s do
               evaluate fitness for s;
6
7
              if fitness for s has decreased significantly
              then
                  set individual fitness to 0;
                  proceed to next individual;
10
              end
          end
11
12
          evaluate fitness for s';
          set individual fitness to fitness for s';
13
14
      produce new population from existing one;
15
16 end
```

Algorithm 2: Reconciling existing skills to body changes made for new skill s'.

```
1 foreach existing skill s do
2 | foreach generation do
3 | foreach individual in the population do
4 | mutate control for skill s;
5 | evaluate fitness for s;
6 | set individual fitness to fitness for s;
7 | end
8 | produce new population from existing one;
9 | end
10 end
```

Results

The experiments demonstrate the advantages of the continuing morphological evolution enabled by the extended ESP algorithm. In the first subsection (*Strike Results*), an experiment from the original ESP system is reproduced in the extended ESP system, with dramatically different results. In the second subsection (*High-Reach Results*), a learning challenge designed to highlight the extended system's advantages is presented, and detailed benefits are described. Note that, while extended ESP maintains original ESP's ability to construct complex hierarchical behaviors, that ability is inherited largely without modification in the new system. Therefore, the experiments in this paper are used instead to demonstrate the extended system's success in more challenging applications that were impossible in the original system. Video illustrating both of these result sections can be viewed online.³

Strike Results

An important part of the original ESP system's primary experimental result was to add a strike behavior to a locomoting creature (toward the larger goal of developing a complex fight-or-flight behavior). In this section, that portion of the old experiment is reproduced in the extended system, and a broad range of novel strategies and morphological changes is observed.

Strike in Original ESP Figure 1a depicts a creature evolved for locomotion in the underlying EVC system (which is common to both the original and extended versions of ESP). In original ESP, that creature consistently solved the challenge of producing a striking behavior by using its existing skeletal structure to either jump up and down or smash the ground with its limbs, without any opportunity to explore the potential for new strategies or better adaptation that might result from continuing full morphological development.

Strike in Extended ESP When the morphology is allowed to continue to evolve, however, new strategies become possible, and even old strategies may be better executed with morphological changes adapted to their specific needs. The extended ESP system develops a variety of such solutions, as can be seen in Figures 1b through 1f.

High-Reach Results

The second experiment was designed to highlight the benefits of the extended ESP system over the old implementation. Specifically, a selection of three different locomoting creatures was evolved to learn the additional skill of reaching for a high target, and the subsequent differences of results for the original and extended ESP implementations were examined in detail. The extended ESP system led to two types of improvements: 1) increased diversity of results, and 2) increased numerical fitness.

³http://youtu.be/fyVr7gdGEPE

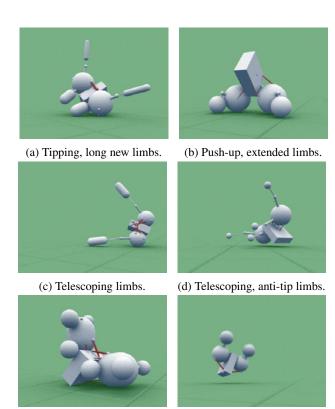


Figure 7: Greater variety through continued morphology evolution. The locomoting creature of Figure 1a was further evolved using the extended ESP system to adapt to a high-reach task. The results demonstrate the potential of continued morphology evolution to produce a great degree

(f) Jump, swing extensions up.

(e) Tip with enlarged limbs.

of useful variety.

Greater Variety The locomoting creature of Figure 1a was evolved toward the new high-reach goal, using both the original and the extended ESP implementation.

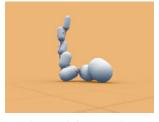
In the original ESP system, only two strategies were observed, within which the results were extremely uniform. Using morphology unchanged from the original locomotion result, all such creatures developed to either jump as high as possible, or reach a limb up by tipping over onto the other limb. In both cases, the results were limited by the inability of skeletal morphology to adapt to this new task.

In the extended ESP system, in contrast, a wide variety of results was observed, in which a number of novel strategies were used, often to great effect. These solutions are illustrated in Figure 7 (a) through (f).

Better Fitness Another successful solution to the locomotion task produced by the original ESP is shown in Figure 8a. This snake-like creature achieved a high reach by extending one end of its long morphology, while the rest of the body maintained balance.

Extended ESP improved upon this creature by changing



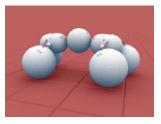


(a) Original ESP result.

(b) Result in new ESP.

Figure 8: Improved fitness via continued morphology evolution. These results demonstrate how the extended ESP system (b) can produce better fitness values (i.e., a higher reach) than the original ESP system (a) by allowing the addition of new body segments.

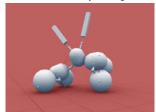
its morphology for the secondary task, while its strategy remained unchanged (Figure 8b). It grew an additional body segment that enabled the higher reach, while allowing it still to perform locomotion to acceptable standards.



(a) Initial locomoting creature.

(b) Subtle body changes.





(c) More obvious body changes.

(d) Dramatic changes in morphology.

Figure 9: Greater variety and improved fitness. The initial locomoting quadruped (a) is evolved for high reach in the extended ESP system (b)-(d). Through a variety of strategies, each of the extended ESP creatures scores better on this new task than any creature from the original ESP system.

Greater Variety and Better Fitness The relatively complex quadruped seen in Figure 9a was a third type of solution developed by the underlying EVC system for the locomotion task. In continued evolution of the high-reach task in the original ESP system, this creature's results were again extremely uniform in approach and fitness. They all reached up with a single limb, and all with approximately equal success. In the extended system, the ability to continue to adapt morphology to this new task led to a diverse set of useful

results, all of which were also more fit than those produced with the original technique.

For example, Figure 9b depicts a creature that pursues the same strategy as the creature in Figure 9a, yet does so more effectively due to subtle morphological adaptations. In Figure 9c, more obvious morphological adaptations have been added to further exceed the uniform performance limit experienced by this creature in the original system, while still employing the same basic technique. In Figure 9d, even more dramatic changes to morphology provide a new way of solving the high reach: This creature employs a new pair of tall, dedicated limbs to even further exceed the performance of the original ESP system.

Discussion and Future Work

Although the extended ESP algorithm has removed the original system's explicit limitations on body changes after the first skill, development of morphology throughout the acquisition of complex skills is still not fully general and completely unlimited. First, the retesting requirements would make morphological development impractical if continued through too many steps of leaf skill addition. To mitigate this issue in the future, it may be possible to do the retesting periodically rather than universally, and run the tests in parallel. Also, the more leaf skills there are, the more likely it is that the morphological change required by one skill is harmful to the others. This limitation is more difficult to overcome, and indeed it reflects the conflicting demands that any creature faces when dealing with complex environments.

In principle, it would be desirable to continue morphology evolution throughout all skill adaptations—not just for those skills that are leaves in the syllabus graph. In this manner, it might be possible to develop morphologies that make transitions between skills easier—for example by making the creature more stable or more agile. To make such evolution possible without increasing testing too much, a rolling horizon of "leaf" skills that travels through the syllabus hierarchy might be implemented. The idea is that once a lower skill will no longer be explicitly required for any subsequent skills, it need not be retested or maintained at all. With a well-chosen sequence of skill learning, an approximately constant-sized wave of "leaves" might result, sweeping gradually through the hierarchy.

Another potential area for future study that might be possible with extended ESP is to increase the morphological complexity of EVCs. While it was not investigated in this paper, the ability of the body to embody multiple types of physical intelligence simultaneously could ultimately lead to a greater degree of physical complexity, and thereby more interesting and capable EVCs.

Conclusion

This paper described an extension of the original ESP system to continue adaptation of morphology beyond the initial

skill, while still incrementally producing high-complexity behaviors. The benefits of this continued adaptation were demonstrated through experiments in which the extended ESP system generated a greater variety of solutions and solutions with higher fitness. Therefore, in the future, this method should make it possible to build EVCs with complex and believable behavior.

Acknowledgements

This research was supported by NSF grants DBI-0939454 and IIS-0915038, NIH grant R01-GM105042, and Intel's Visual Computing Program.

References

- Gomez, F. and Miikkulainen, R. (1997). Incremental evolution of complex general behavior. Adaptive Behavior, 5:317–342.
- Lehman, J. and Stanley, K. (2011). Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 211–218. ACM.
- Lessin, D., Fussell, D., and Miikkulainen, R. (2013). Open-ended behavioral complexity for evolved virtual creatures. In Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference, GECCO '13, pages 335–342, New York, NY, USA. ACM.
- Lessin, D., Fussell, D., and Miikkulainen, R. (2014). Trading control intelligence for physical intelligence: Muscle drives in evolved virtual creatures. In *Proceeding of the Sixteenth Annual Conference on Genetic and Evolutionary Computation Conference*, GECCO '14, New York, NY, USA. ACM.
- Lipson, H. and Pollack, J. B. (2000). Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974–978.
- Miconi, T. (2008). In silicon no one can hear you scream: Evolving fighting creatures. *Genetic Programming*, pages 25–36.
- Mitchell, M. (1998). An Introduction to Genetic Algorithms. MIT Press, Cambridge, MA, USA.
- Pilat, M. L. and Jacob, C. (2010). Evolution of vision capabilities in embodied virtual creatures. In *Proceedings of the 12th* annual conference on Genetic and evolutionary computation, GECCO '10, pages 95–102, New York, NY, USA. ACM.
- Selfridge, O. G. (1958). Pandemonium: a paradigm for learning in Mechanisation of Thought Processes. In *Proceedings of a Symposium Held at the National Physical Laboratory*, pages 513–526, London. HMSO.
- Shim, Y. and Kim, C. (2003). Generating flying creatures using body-brain co-evolution. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 276–285. Eurographics Association.
- Sims, K. (1994). Evolving virtual creatures. In *Proceedings of the* 21st annual conference on Computer graphics and interactive techniques, SIGGRAPH '94, pages 15–22, New York, NY, USA. ACM.
- Urzelai, J., Floreano, D., Dorigo, M., and Colombetti, M. (1998). Incremental robot shaping. *Connection Science*, 10:341–360.

Evolving Morphologies with CPPN-NEAT and a Dynamic Substrate

Daniel Richards and Martyn Amos

School of Computing, Mathematics & Digital Technology, Manchester Metropolitan University D.Richards@mmu.ac.uk

Abstract

Recent advances in fabrication technologies open up exciting opportunities to manufacture entirely new types of physical materials and structures. These have varied and specific mechanical properties, which can be exploited in a number of engineering applications, and a growing area of research concerns the generation of two- and three-dimensional designs using these materials. However, the computational tools required to explore large spaces of possible 2-D and 3-D morphologies remain underdeveloped. State-of-the-art evolutionary approaches such as CPPN-NEAT HyperNEAT-LEO are often used to explore possible 2-D and 3-D designs, but their ability to construct efficient solutions for practical use in engineering domains remains in question. In this paper, we present an extension of CPPN-NEAT, in which nodes grow connections across a dynamic substrate, and illustrate this by creating efficient 2-D truss structures. Using four benchmark problems, we then demonstrate that our CPPN-NEAT model outperforms HyperNEAT methods for approximating specific connectivity patterns, and suggests important clues regarding how to best harness generative and developmental representations to build scalable and high-performance physical morphologies.

Introduction

Processes of evolution and development have shaped a vast array of physical structures. The desire to construct artificial structures with similar levels of complexity and efficiency is a driving force behind much of contemporary engineering. Recent advances in manufacturing technologies, specifically additive manufacturing (i.e. 3-D printing), open up the very real possibility of fabricating high-performance, physical designs that exhibit complex bio-inspired morphologies and behaviors [1-3]. Critically, this enhanced ability to control how and where material is distributed within structures enables the construction of entirely new classes of materials and objects, for use in various engineering domains. For example, at small scales, bone tissue scaffolds can be fabricated with bioactive glass to exhibit specific mechanical properties, such as high resistance to fracture [4]. At larger scales, these advances suggest vast potential for applications such as flexible cellular microstructures for prosthetic limbs [5], morphing wing designs for aerospace applications [6], and next-generation architectural designs [7,8].

While new fabrication hardware is facilitating exciting opportunities for engineering domains, the computational tools needed to fully exploit these technologies and aid

discovery of functional morphologies with novel mechanical properties, remain relatively underdeveloped.

As noted by Hiller and Lipson [9], well-known structural optimization algorithms, such as homogenization techniques [10], can already successfully address simple design problems such as 2-D and 3-D truss structures. However, because they rely on prior knowledge of how to exploit local gradient information, they are limited in their ability to discover complex designs that meet higher-level *functional* goals.

To address this limitation, evolutionary algorithms are often used to explore large design spaces and generate efficient solutions. Evolutionary methods are potentially useful because they can explore search spaces when no problem specific knowledge exists. However, evolutionary algorithms have their own set of limitations when applied to engineering domains. These include: lack of scalability [11], limited ability to ensure buildable solutions [12], inability to guarantee global optima [13] and difficulty in applying them to design exploration (i.e. beyond late-stage parameter optimization) [14].

Various work based on the NEAT (Neuroevolution of Augmenting Topologies) model (specifically, CPPN-NEAT [15, 24-26] and HyperNEAT [27, 28]) suggests the existence of exciting opportunities for addressing many of these issues, critically offering key advantages in terms of scalability. However, there still exist significant challenges relating to the use of these approaches for building 2-D and 3-D morphologies for engineering domains. For example, Devert et al [20] have recently highlighted critical "approximation accuracy" limitations of HyperNEAT when applied to 2-D truss structure optimization. Additionally, Fenton et al [32] note that ANN-based methods are less desirable than grammatical systems, due to the difficulties of imposing physical constraints.

In this paper, we precisely address these two major issues and present an approach that combines NEAT with Compositional Pattern-Producing Networks (CPPN). Specifically, this approach exploits growth and relative targeting of connections on a dynamic substrate to: (A) create 2-D truss structures that out-perform those produced by similar HyperNEAT approaches on four well-known topology optimization benchmark problems, and (B) enable simple ways of embedding problem-specific constraints for engineering applications. We first review related work, and then present our model, demonstrating how it extends a typical CPPN-NEAT method. We describe our experimental setup and present our results, before concluding with a discussion and suggestions for further work.

Background

Generative and developmental representations offer a powerful framework for creating 2-D and 3-D designs [16,17]. Kicinger et al [18], present a generative representation, based on cellular automata, to evolve tall steel structures for building design. Kowaliw et al [19] use a novel embryogeny to evolve 2-D truss structures. Recently, Devert et al [20] demonstrate a novel ontogenic approach for 2-D truss optimization. For a comprehensive review of evolutionary structural design methods, see [21] and [22].

Recent work, particularly within the area of evolutionary robotics, demonstrates the construction of diverse 2-D and 3-D morphologies by evolving CPPNs [15] with Stanley and Miikkulainen's NEAT algorithm [23]. For example, Clune and Lipson [24] use CPPN-NEAT to evolve 3D printed objects; Cheney et al [25], evolve virtual creatures with novel morphologies comprising multiple materials; Hiller and Lipson [9], evolve multi-material physical objects to meet high-level functional goals, such as specific deformations of 3-D beams; Auerbach and Bongard [26] evolve virtual creatures with diverse locomotive behaviors; and Szerlip and Stanley [27] use HyperNEAT-LEO [28,29] to evolve diverse, functional 2-D designs for the simulation engine *Sodarace*.

We suggest that NEAT-based approaches offer significant potential for use in engineering domains, by enabling the discovery of novel material compositions with *specific mechanical properties* and *high-level functionality*. However, as highlighted by Devert et al [20], HyperNEAT currently has problems with accurately generating specific connectivity patterns, and this limitation makes it difficult to optimize even relatively simple 2-D truss structures. At the core of this issue is that HyperNEAT struggles to produce *modular* networks. Verbancsics and Stanley [29] show that by adding a bias towards creating local connections, HyperNEAT-LEO can create modular solutions and improve performance on various control problems. However, we believe that this approach, in its current form, is not enough to create efficient 2-D and 3-D morphologies – we support this claim in the Results section.

The second key challenge for CPPN-NEAT, when applied to engineering problems, is the inclusion of *physical constraints*. For example, to apply standard structural analysis techniques, solutions must have all parts suitably connected (i.e. no disconnected elements), yet dealing with fully connected solutions can be computationally expensive. For this reason, grammatical systems are often the favored generative approach to evolving buildable 2-D and 3-D solutions. Notably, Hornby and Pollack [30] evolve L-System grammars to create buildable table designs; Rieffel and Smith [31] use a grammatical approach to grow soft-bodied robot morphologies; and Fenton et al [32] use grammatical evolution to optimize simple truss structures and restrict the design space to a set of standard member sizes.

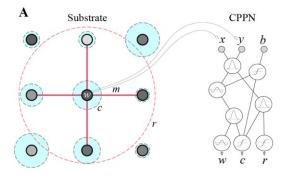
The key insight in this paper is that CPPN-NEAT can control simple (grammar-like) growth rules that play out on a 2-D (or 3-D) grid and generate local connectivity patterns, which are interpreted as physical designs. The benefit of this extra layer of abstraction is that we can: (A) more accurately approximate specific connectivity patterns; (B) easily enforce physical constraints and (C) exploit CPPN-NEAT's scalability and capacity to create geometric regularities.

Methods

Representation

CPPN-NEAT and HyperNEAT have already been described in detail [15, 24-29, 33], so we provide only a brief summary, and mainly focus on how our proposed method differs from previous work. CPPNs are similar to neural networks, yet nodes may contain a variety of different mathematical functions and can be evolved using NEAT [23]. CPPN-NEAT can be used to evolve complex geometric patterns. For example, 2-D patterns can be drawn by querying a CPPN and setting the color of each pixel on a canvas as a function of its x and y coordinates. Similarly, HyperNEAT uses CPPNs to specify the weight, w, of all connections in a larger fully connected feed-forward neural network (termed the "substrate", to distinguish it from the CPPN, which is also a network), by querying the coordinates of each input node i and output node j. That is, for each connection, the CPPN inputs x_i , y_i , x_j , y_i and outputs w.

The key idea in this paper is to use CPPNs to create 2-D morphologies in a similar manner to HyperNEAT (as [27]), but with *one key difference*. That is, instead of querying a fully connected *fixed* substrate of possible connections and specifying the dimensions of individual truss members, we use CPPNs to generate growth instructions for each node as a function of its x and y coordinates (Fig 1).



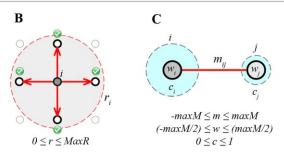


Figure 1. Growth of structural elements. (A) Nodes are initialized with growth instructions by querying a CPPN. Nodes grow connections using three properties: r, w, c. (B) Each node has a range of influence, r, and is permitted make connections between all other nodes which are within this radius. (C) Nodes also have a weight, w, and a concentration, c. Connections between nodes i and j have a cross-sectional area, m_{ij} , which is created by summing w_i and w_j , with a potential bias that is determined by concentrations c_i and c_i .

The shift from directly querying connections (specific targeting) to querying nodes and then growing connections (relative targeting of connections) provides at least three interesting possibilities when generating network structures. Firstly, it imposes a hard-coded bias towards creating local connections. That is, unlike [29], in this approach the ability to create longer connections actively increases the number of possible connections in the entire structure, thus enlarging the dimensionality of the problem. This means that networks with shorter, local connections are often easier to optimize, and thus more likely to emerge. Secondly, fixed fully connected substrates can be replaced by dynamic substrates, where the maximum connectivity of any individual node varies across the substrate as a function of growth. Notably, the bottleneck in many engineering problems is the computation time required for evaluation, so eliminating the need to simulate fully connected substrates can result in significant savings of CPU time. Thirdly, as we have previously shown [34], growth provides a useful mechanism for imposing necessary physical constraints. For example, during growth, individual nodes can be limited to a maximum or minimum number of connections.

As shown in Figure 1, nodes are initialized by querying a CPPN (using Cartesian coordinates as inputs), and then grow connections on a larger substrate, where: $-1 \le x \le 1$ and $-1 \le y \le 1$. The CPPN uses three inputs (x and y coordinates and one bias) and returns three output values (w, c, r). Each connection has a cross-sectional area, m, which is defined by summing the weight, w, from each endpoint node and potentially applying a small bias based on the concentration value, c, of both end nodes. Thus, to define the cross-sectional area m_{AB} we use:

bias =
$$(max(c_A, c_B) - min(c_A, c_B)) \times Q$$

if $(c_A \ge 0.5 \text{ and } c_B < 0.5)$
 $m_{AB} = (w_A \times (1 + bias)) + (w_B \times (1 - bias))$
else if $(c_A < 0.5 \text{ and } c_B \ge 0.5)$ (1)
 $m_{AB} = (w_A \times (1 - bias)) + (w_B \times (1 + bias))$
else $m_{AB} = w_A + w_B$

Where m_{AB} is the cross-sectional area of the connection between nodes A and B, c is the concentration value, w is the weight of each node, and Q in a fixed coefficient value used to determine the maximum bias. In this paper, we set Q=0.2 for all problems.

Each node has a minimum and maximum range, r. To avoid matrix errors during structural analysis, we enforce a minimum degree of connectivity for all nodes (Fig 2B).

There are two significant implications of this; firstly, nodes can be completely "switched off" by reducing the r below the minR threshold. This means that no other nodes can connect, and this provides a useful mechanism for sculpting network topology. Secondly, because nodes contain useful information about their immediate neighborhood, null connections that do not contribute to the node's minimum connectivity are easily eliminated, excluding them from computationally expensive analysis. For example, fully connected solutions (Fig 2C), where all members have null cross-sectional areas ($m \le 1.e^{-7}$), can be converted into Figure 2B before evaluation, which is more computationally efficient to simulate. Note, if minimum connections are defined as "null", we use a very small cross-sectional area ($m = 1.e^{-7}$), as is standard in truss optimization.

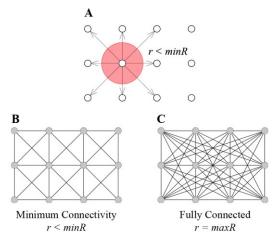


Figure 2. Enforcing minimum and maximum connectivity. (A) Each node must at least connect to all of its immediate neighbors. This distance is defined as the minimum connectivity limit: minR. If a node's r < minR, all connections made to this node become null and it is connected to its immediate neighbors. (B) Minimum connectivity of all nodes. (C) Fully connected structure.

Truss structures (as shown in Fig 2) must not contain overlapping (duplicate) connections; therefore to enable a dynamic substrate, further constraints must be applied to ensure that such connections are disallowed. To achieve this, we modify our "data-tag" approach used during node interactions, described in detail in [34]. As in [34], nodes swap and manage a small set of data-tags to keep track of their existing and new connections. Data-tags are checked before new connections are made, to avoid creating duplicates. Additionally, to avoid overlapping connections, we constrain nodes so that they are only permitted to connect to neighbors when there are no intermediate nodes.

Static Analysis

To evaluate the 2-D structures we use the linear *direct* stiffness method [35], which is a common finite element method (FEM). The process involves modeling the stiffness properties of each truss member and using the information to assemble a larger global stiffness matrix, K(a), which describes the mechanical behavior of the entire structure:

$$K(a) = \sum_{j=1}^{n} a_j K_j$$

Where a_j is the cross-sectional area of the *jth* truss element and K_j is the member stiffness of the *jth* truss element. When M forces $(f_1,...,f_M)$ are imposed on nodes in the truss they react and move. The displacement, u_j of loaded nodes is solved with the linear system:

$$K(a)u_k = f_k, \qquad k = 1, \dots, M$$

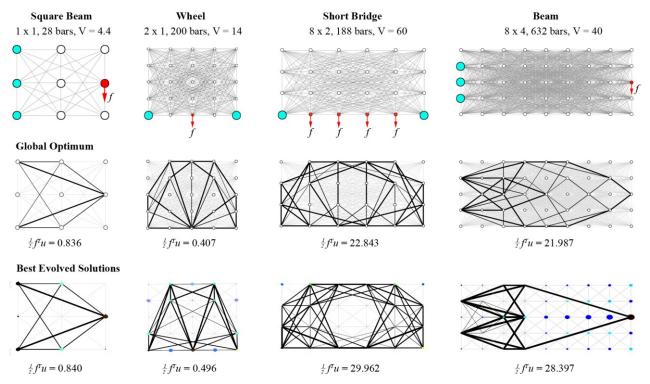


Figure 3. Benchmark Problems. (Top row) Four benchmark problem instances. Blue nodes have restricted degrees of freedom, and red nodes have an imposed one unit load in the direction of the arrow. The descriptions above give (a) x and y dimensions of truss (note: figures show the *substrate*, not the absolute x and y dimensions of each truss as used in FEM analysis), (b) number of bars and (c) the volume constraint, V (see [36] for further details). (Middle row) Global optima (from [36]). (Bottom row) Best evolved solutions using CPPN-NEAT with dynamic substrate. Here node color and size represent the node properties c and r, respectively.

Benchmark Problems

We test our method using the same truss benchmark problems as those used by Devert et al [20] (Fig 3), and compare our results with similar HyperNEAT methods.

The benchmark problems are relatively simple, have known global optima [36], and can be solved perfectly using evolutionary approaches with direct representations. However, as highlighted by Devert et al [20], HyperNEAT (HyperNEAT 3.0 C++ package with default parameters) struggles to generate good solutions to these problems. They argue that, due to this limitation, ontogenic representations are superior. However, we suggest that such ontogenic representations are only applicable when problems have valuable gradient information that is already known to be exploitable. For example, Devert et al's ontogenic encoding uses 32 FEM calls per solution and works by iteratively querying a CPPN, each time inputting the structural strain of each truss member and returning an incremental modification of the cross-sectional area. This technique does indeed provide superior solutions to similar generative representations. However, when such gradient information is readily available and the correlation between parameters is well understood (i.e. if member strain is low, reduce cross-sectional area), existing homogenization techniques are generally orders of magnitude faster [10, 37].

Since our motivation is to explore large search spaces where useful gradient information is largely unknown, and homogenization techniques do not exist, we argue that better generative representations will provide a valuable way of discovering complex material compositions in various problem domains [1-9]. Consequently, we compare our method against: (1) Devert et al's recorded HyperNEAT results [20], (2) HyperNEAT with a locality seeded LEO [29], and (3) our model without the dynamic substrate, where r is binary either "on" with full connectivity, or "off" (r < minR) with all null connections (see Fig 2).

We use the common topology optimization objective:

$$\min \frac{1}{2} f^T u$$

Subject to the constraints:

$$\sum_{j=1}^{n} a_j l_j \le V$$

$$a \in [0.1]$$

Where: a_j is the cross-sectional area (a continuous value between 0 and 1) of the *jth* bar, l_j is the length of the *jth* bar, and V is a problem specific volume constraint (see Fig 3).

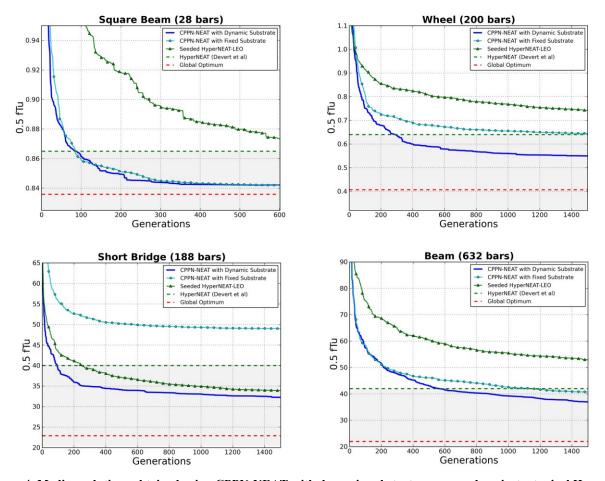


Figure 4. Median solutions obtained using CPPN-NEAT with dynamic substrate, compared against a typical HyperNEAT method (indiated as a fixed threshold value, showing the median best solutions obtained by [20] over an equivalent number of FEM evaluations, from 64 independent runs). HyperNEAT with a seeded LEO [29], our CPPN-NEAT with fixed substrate, and the known global optimum (taken from [36] – indicated as a dotted line). On all problems, CPPN-NEAT with a dynamic substrate performs best.

Experimental Details

For all benchmark problems, we perform 20 independent runs, each with a population of 150 CPPNs, evolved for a maximum of 1500 generations. As NEAT is a maximization algorithm, we adapt the previously outlined objective function to define our fitness function:

$$max \frac{1}{\frac{1}{2}f^T u}$$

We use our own java implementation of NEAT and HyperNEAT-LEO. CPPNs use the following activation functions: *Gaussian, Cosine and Sigmoid* with equal probability of being produced. We promote 25% of the population with elitism, and there is an 80% chance of mutating individuals after crossover.

Mutation rates are 0.03, 0.05 and 0.8 for adding a new node, adding a new link and perturbing connection weights, respectfully – and probability of interspecies mating is 0.001. We use a dynamic compatibility threshold, where the target number of species is 10 and the niche size required for elitism is 5. Finally, we set compatibility coefficients to: $c_1 = 2.0$, $c_2 = 2.0$, $c_3 = 1.0$.

The Young's modulus of all non-null beams is 1.0 and for null members is 0.0. All force loads (shown with a red arrow in Fig 3) have a magnitude of 1.0 and are in the direction of the arrow. Each problem has a maximum volume of material, V, if any solution has a *volume* > V, the cross-sectional area of all members are linearly scaled to meet V. The minimum cross-sectional area of any truss member is $1.0e^{-7}$ and anything below this value is null. Finally, for all of these experiments, the maximum cross-sectional area (maxM) is 1.0.

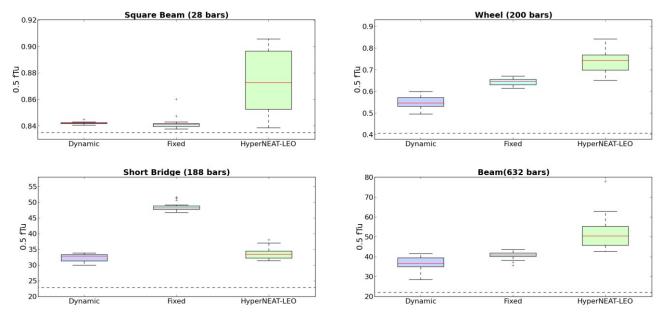


Figure 5. Best solutions obtained for different problems over 20 independent runs for CPPN-NEAT with dynamic substrate, CPPN-NEAT with fixed substrate and the seeded HyperNEAT-LEO. Dotted lines indicate global optima.

Results

Our results demonstrate that CPPN-NEAT with dynamic substrate outperforms both standard HyperNEAT and seeded HyperNEAT-LEO on all four benchmark problems. Specifically, the dynamic property of the substrate, coupled with the ability to silence nodes during growth (i.e. when r < minR, see Fig 2), enables our CPPN-NEAT method to create better truss solutions and more closely approximate specific connectivity patterns of the known global optimum (Fig 3).

Figure 4 shows the median solutions obtained using our dynamic method in comparison to: (A) standard HyperNEAT (as described in [20]), (B) HyperNEAT-LEO with a seeded bias towards expressing local connections [29], and (C) our CPPN-NEAT method using a fixed and fully connected substrate. As shown, the CPPN-NEAT with dynamic substrate converges faster than all other methods and quickly improves on the best solutions found by [20]. Figure 5 shows the best solutions obtained with each approach; we see that CPPN-NEAT with dynamic substrate improves on the best median solutions generated with the seeded HyperNEAT-LEO by 3.5% on the smallest *square beam* problem, a more substantial 26.5% on the *wheel*, 2.8% on the *short-bridge* and 27% on the largest *beam* problem.

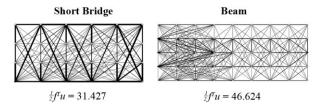


Figure 6. Best truss designs obtained with the seeded HyperNEAT-LEO over 20 independent runs.

Interestingly, the seeded HyperNEAT-LEO performs well on the *short-bridge* problem and significantly outperforms standard HyperNEAT and CPPN-NEAT with fixed substrate. We attribute this increased performance to the substrate configuration of the problem. That is, the short-bridge problem uses a 5 x 3 grid of nodes, yet the x and y dimensions of the substrate of the are always equal $(-1 \le x \le 1 \text{ and } -1 \le y$ $\leq I$). This means that, unlike the square-beam and wheel problem, nodes are not equally spaced in the substrate, so here the HyperNEAT-LEO with seeded bias for creating local connections is able to improve solutions. However, as shown in Figures 4 and 5, the seeded HyperNEAT-LEO actually performs much worse on the larger beam problem, which has similar (8 x 4) substrate configuration. To understand this difference, we can look at the type of truss designs produced by seeded HyperNEAT-LEO (Fig 6). Here we see that a key problem is that it struggles to eliminate non-essential connections, and instead tends favor highly connected designs. This tendency becomes increasingly detrimental to truss optimization as the problems scale up and we increase the number of possible connections.

Critically, CPPN-NEAT with dynamic substrate *does not* suffer from this problem, and, in contrast, evolves solutions with much more clearly differentiated network morphologies (Fig 3). Interestingly, we observe that during the early stages of evolution, CPPN-NEAT with dynamic substrate tends to create truss structures with minimal connectivity patterns that become more complex over time (similar to NEAT) as structurally significant nodes in the substrate become densely connected and establish longer connections.

These results suggest that while seeding HyperNEAT-LEO with a bias towards creating local connections is clearly a valuable method for improving performance and modularity in ANN problem domains [39], when creating 2-D (and most likely 3-D) designs, CPPN-NEAT with dynamic substrate provides superior solutions.

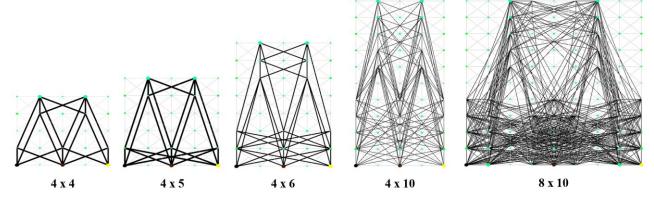


Figure 7. Scaling trusses without further evolution. The physical dimensions and substrate resolution (number of nodes in x and y axis) can be manually manipulated following evolution, and trusses retain many evolved characteristics.

Discussion

We present a novel extension of CPPN-NEAT for evolving functional 2-D network structures. We show that CPPN-NEAT with a dynamic substrate outperforms similar state-of-the-art HyperNEAT methods on four well-known benchmark problems for topology optimization, and more accurately approximates specific connectivity patterns of known global optima.

We acknowledge that none of the CPPN-based methods presented here are able to find the global optimum for these problems, unlike state-of-the-art evolutionary strategies, which do converge to global optima (as shown by Devert et al [20]). However, methods that rely on *direct* representations are prone to issues of (lack of) scalability, and are thus limited to addressing relatively simple design problems. Generative encodings will benefit various engineering domains [1-9] and ultimately allow us to exploit advanced manufacturing technologies and capabilities (such as high-value functionally graded solutions, multi-material composites, and so on.) In this context, "optimality" is less important than the ability to obtain new types of morphology that exhibit some specific functional behavior(s). CPPN-NEAT with dynamic substrate may hold important clues towards achieving this goal, and here we have presented an early proof-of-principle (albeit on a simple benchmark problem). In this concluding section, we describe additional properties of CPPN-NEAT with dynamic substrate that will offer useful trajectories for further research.

The key feature of our model is a more "biologically plausible" method of creating connectivity patterns. Instead of using HyperNEAT to define properties of specifically identified connections, we use a CPPN to control an additional network growth process. During this process, nodes use relative positional information to target surrounding nodes and grow locality biased connectivity patterns. In this paper, our growth step occurs just once to build 2-D truss designs. However, if we were to set our model within a physics-based simulation environment, nodes could continue to build and destroy connections over the lifetime of the structure, as nodes move. Such an approach could provide interesting *locomotive* behaviors that may be extremely robust to perturbation.

A second interesting implication of creating connections through growth is that we define new opportunities to exploit *node-based information*. For example, in the benchmark problems used in this paper, nodes are explicitly encoded with information that is used to (A) restrict degrees of freedom and (B) impose physical loads. In many engineering problems, this type of information is known well before optimization takes place, and could therefore easily be fed into CPPNs as additional information. Notably, Clune et al [24], have previously demonstrated that feeding additional information, relating to distance to center of a 3-D grid of voxels, can be exploited to create more rounded geometric features in solutions. Consequently, the ability to utilize information about boundary conditions may be useful in future work.

Thirdly, an exciting property of CPPNs is that the solutions they encode effectively obtain *infinite resolution*. For example, 2-D pictures evolved with CPPNs never get pixelated when they are scaled up. Similarly, truss designs evolved with CPPN-NEAT can be manipulated following evolution, and retain many of their characteristics (Fig 7). This property could have at least two useful applications. Firstly, the ability to increase resolution during evolution may help to optimize large-scale truss and/or network-based structures. Secondly, this added flexibility may provide new possibilities for *interactive* evolution, allowing network-based solutions to be explored and scaled in real-time. Indeed, in our current Java-based method, users can extract parameters of the underlying CPPN (connection weights) and manually adjust evolved solutions in real-time.

In addition to these avenues of enquiry, we will investigate the dynamical behavior of our model, in order to better understand (and improve) the method. In particular, we anticipate significant future savings in computational effort as we scale up the size of the substrate, due to our method avoiding non-essential connections between nodes. Future work will rigorously investigate the absolute performance improvement that may be obtained with our model on non-trivial problem instances.

Finally, further work is now underway to explore more complex 3-D problems with non-trivial mechanical properties (such as compliant mechanisms). Notably, the method presented in this paper can easily be extended to 3-D design by simply adjusting the FEM solver (see Fig 8).

References

- [1] Dimas, L. S., Bratzel, G. H., Eylon, I., & Buehler, M. J. (2013). Tough composites inspired by mineralized natural materials: computation, 3D printing, and testing. *Advanced Functional Materials*, 23(36), 4629-4638
- [2] Cranford, S., & Buehler, M. J. (2010). Materiomics: biological protein materials, from nano to macro. *Nanotechnology, Science and Applications*, 3, 127.
- [3] Lipson, H., & Kurman, M. (2013). Fabricated: The New World of 3D Printing. John Wiley & Sons.
- [4] Fu, Q., Saiz, E., Rahaman, M. N., & Tomsia, A. P. (2011). Bioactive glass scaffolds for bone tissue engineering: state of the art and future perspectives. *Materials Science and Engineering: C*, 31(7), 1245-1256.
- [5] Abdelaal, O. A., & Darwish, S. M. (2012). Analysis, Fabrication and a Biomedical Application of Auxetic Cellular Structures. *IJEIT*, 2(3), 218-223.
- [6] Chu, C., Graf, G., & Rosen, D. W. (2008). Design for additive manufacturing of cellular structures. *Computer-Aided Design and Applications*, 5(5), 686-696.
- [7] Oxman, N. (2011). Variable property rapid prototyping: Inspired by nature, where form is characterized by heterogeneous compositions, the paper presents a novel approach to layered manufacturing entitled variable property rapid prototyping. *Virtual and Physical Prototyping*, 6(1), 3-31.
- [8] Menges, A. (2012). Material computation: Higher integration in morphogenetic design. *Architectural Design*, 82(2), 14-21.
- [9] Hiller, J., & Lipson, H. (2012). Automatic design and manufacture of soft robots. *IEEE Transactions on Robotics*, 28(2), 457-466.
- [10] Bendsøe, M. P., & Kikuchi, N. (1988). Generating optimal topologies in structural design using a homogenization method. Computer Methods in Applied Mechanics and Engineering, 71(2), 197-224.
- [11] Stanley, K. O., & Miikkulainen, R. (2003). A taxonomy for artificial embryogeny. *Artificial Life*, 9(2), 93-130.
- [12] Rieffel, J., & Pollack, J. (2005). Automated assembly as situated development: using artificial ontogenies to evolve buildable 3-d objects. *Proceedings of GECCO 2005*, ACM, 99-106.
- [13] Sigmund, O. (2011). On the usefulness of non-gradient approaches in topology optimization. *Structural and Multidisciplinary Optimization*, 43(5), 589-596.
- [14] Shea, K., Aish, R., & Gourtovaia, M. (2005). Towards integrated performance-driven generative design tools. *Automation in Construction*, 14(2), 253-264.
- [15] Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. Genetic Programming and Evolvable Machines, 8(2), 131-162.
- [16] Doursat, R., Sayama, H., & Michel, O. (2012). Morphogenetic Engineering: Toward Programmable Complex Systems. Springer.
- [17] Kumar, S., & Bentley, P. J. (Eds.). (2003). On Growth, Form and Computers. Academic Press.
- [18] Kicinger, R., Arciszewski, T., & De Jong, K. (2004). Morphogenesis and structural design: cellular automata representations of steel structures in tall buildings. *Proceedings of Evolutionary Computation*, 2004. CEC2004. Vol. 1, 411-418.
- [19] Kowaliw, T., Grogono, P., & Kharma, N. (2007). The evolution of structural design through artificial embryogeny. *Proceedings of Artificial Life*, 2007. IEEE Symposium on ALIFE'07, 425-432.
- [20] Devert, A., Weise, T., & Tang, K. (2012). A study on scalable representations for evolutionary optimization of ground structures. *Evolutionary computation*, 20(3), 453-472.
- [21] Kicinger, R., Arciszewski, T., & Jong, K. D. (2005). Evolutionary computation and structural design: A survey of the state-of-the-art. *Computers and Structures*, 83(23), 1943-1978.
- [22] Zavala, G. R., Nebro, A. J., Luna, F., & Coello, C. A. C. (2013). A survey of multi-objective metaheuristics applied to structural optimization. Structural and Multidisciplinary Optimization, 1-22.
- [23] Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2), 99-127.

- [24] Clune, J., & Lipson, H. (2011). Evolving 3d objects with a generative encoding inspired by developmental biology. ACM SIGEVOlution, 5(4), 2-12.
- [25] Cheney, N., MacCurdy, R., Clune, J., & Lipson, H. (2013). Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. *Proceedings of GECCO 2013*, 167-174.
- [26] Auerbach, J. E., & Bongard, J. C. (2012). On the relationship between environmental and mechanical complexity in evolved robots. *Artificial Life*, 13, 309-316.
- [27] Szerlip, P., & Stanley, K. (2013). Indirectly Encoded Sodarace for Artificial Life. Advances in Artificial Life, ECAL Vol. 12, pp. 218-225
- [28] Stanley, K. O., D'Ambrosio, D. B., & Gauci, J. (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, 15(2), 185-212.
- [29] Verbancsics, P., & Stanley, K. O. (2011). Constraining connectivity to encourage modularity in HyperNEAT. *Proceedings of GECCO* 2011, pp. 1483-1490.
- [30] Hornby, G. S., & Pollack, J. B. (2001). The advantages of generative grammatical encodings for physical design. *Proceedings of the 2001 Congress on Evolutionary Computation*, Vol. 1, pp. 600-607.
- [31] Rieffel, J., & Smith, S. (2010). A Face-Encoding Grammar for the Generation of Tetrahedral-Mesh Soft Bodies. In *ALIFE*, 414-420.
- [32] Fenton, M., McNally, C., Byrne, J., Hemberg, E., McDermott, J., & O'Neill, M. (2014). Automatic innovative truss design using grammatical evolution. *Automation in Construction*, 39, 59-69.
- [33] Gauci, J., & Stanley, K. (2007). Generating large-scale neural networks through discovering geometric regularities. *Proceedings of GECCO* 2009, 997-1004.
- [34] Richards, D., Dunn, N., & Amos, M. (2012). An evo-devo approach to architectural design. *Proceedings of GECCO 2012*, (pp. 569-576). ACM.
- [35] Felippa, C. A. (2004). Introduction to Finite Element Methods. University of Colorado, Boulder, http://www.colorado. edu/engineering/CAS/courses.d/IFEM.d.
- [36] Achtziger, W., & Stolpe, M. (2007). Truss topology optimization with discrete design variables—guaranteed global optimality and benchmark examples. Structural and Multidisciplinary Optimization, 34(1), 1-20.
- [37] Bendsoe, M. P., & Sigmund, O. (2003). Topology Optimization: Theory, Methods and Applications. Springer.

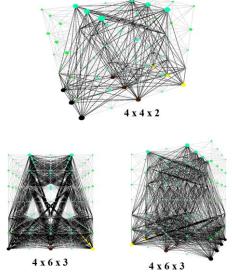


Figure 8. 2-D truss designs can easily be extended into 3-D.

Collective Robotics

Understanding the Role of Recruitment in Collective Robot Foraging

Lenka Pitonakova^{1,2,3}, Richard Crowder ² and Seth Bullock^{1,2}

¹Institute for Complex Systems Simulation and ²School of Electronics and Computer Science
University of Southampton, United Kingdom

³I.pitonakova@soton.ac.uk

Abstract

When is it profitable for robots to forage collectively? Here we compare the ability of swarms of simulated bio-inspired robots to forage either collectively or individually. The conditions under which recruitment (where one robot alerts another to the location of a resource) is profitable are characterised, and explained in terms of the impact of three types of interference between robots (physical, environmental, and informational). Key factors determining swarm performance include resource abundance, the reliability of shared information, time limits on foraging, and the ability of robots to cope with congestion around discovered resources and around the base location. Additional experiments introducing odometry noise indicate that collective foragers are more susceptible to odometry error.

Introduction

Foraging is a well-studied behaviour in both animals and robots. Just as biological species implement foraging strategies that are adapted to their specific niches, it would be desirable to tailor robot foraging strategies to the particular challenges that they face. For example, large animals like wolves, hyenas or lions forage alone when their food is dispersed, while social insects such as ants, bees and termites recruit their nest mates to collectively obtain food from a patch that can be exploited over a number of visits. When should robots forage collectively?

Since collective strategies that rely on communication, co-ordination, interaction, etc., will tend to require more expensive robots and more programming time than independent, individualist robots, it is important for designers to resort to collective strategies only when they significantly improve collective performance.

In this work, the resource gathering performance of a simulated swarm of collective robots that recruit each other to profitable resource locations is compared with that of a swarm of individualists that forage independently. Swarms are evaluated across a range of scenarios that differ in terms of the challenge involved in locating resource deposits, as well as their spatial distribution and variation in terms of volume and quality. By analysing variation in performance

we aim to uncover which environmental properties favour collective foraging approaches in robot swarms.

Background

Most animals, especially carnivores, perform solitary foraging (Gittleman, 1989). For example, forest birds search for insects alone if food is sparse and many insects are needed (Robinson and Holmes, 1982). Frigate birds searching for fish also do so solitarily, despite the fact that they live in colonies. They tend to disperse while searching in order to optimise their probability of success and rarely return to an area where they have previously fed (Weimerskirch et al., 2004). Similarly, hyenas (Holekamp et al., 2012), lionesses (Stander and Albon, 1993) and chimpanzees (Busse, 1978) hunt alone during periods of sparse prey abundance.

By contrast, some creatures forage collectively, recruiting their collaborators to profitable food locations by sharing information. Recruited foraging is most common in social insects and appears either in the form of stigmergy or direct signalling. Stigmergy is utilised by ants (e.g., Beekman, 2001) and termites (Arab et al., 2012) when they lay pheromone trails that lead to food, changing their environment such that it stores useful information that guides the behaviour of recruited conspecifics.

Bees, however, rely on directly influencing their nest mates. Successful foragers return to the nest to perform a waggle dance that communicates both food quality and location (relative to the dance floor) to watching bees (Seeley, 1994; Granovskiy et al., 2012). Flexibility and sophistication is achieved by both scouting for new food sources and re-evaluation of old foraging sites, allowing bee colonies to rapidly adapt to changing flower quality (Seeley, 1992; Biesmeijer and de Vries, 2001)

A number of approaches have been taken to implementing foraging in robot swarms, including random walking robots that forage independently (e.g., Hoff et al., 2010), bucket brigading swarms where each robot is only responsible for its own portion of a foraging area and items travel towards a base location by being passed between robots (e.g., Shell and Mataric, 2006; Lein and Vaughan, 2009), robots that di-

rectly signal to each other about where items can be found (e.g., Rybski et al., 2004; Jevtic and Gazi, 2010; Sarker and Dahl, 2011), and swarms that use stigmergy, where certain robots act as beacons, maintaining an information gradient towards a deposit (e.g., Hoff et al., 2010).

It is notable that robot swarms are often investigated in a single environmental scenario with a specific distribution of target items (typically a uniform random distribution) or in scenarios that are particularly suited to the swarm strategy. An exception to this was the work of Hoff et al. (2010) who compared a group of randomly walking individualists to a swarm that used stigmergy. However, the individualists in these experiments were highly disadvantaged as they had to find the drop off location by random walk every time they acquired a target and thus performed very poorly.

Here we implement an idealised and simplified version of bee-like recruitment in which robots that are near to each other may exchange subjective information about recently visited resource locations. By exploring a range of environments, we aim to discover both the conditions that favour collective foraging and those that do not.

Methods

Simulation environment

Robots of size 10×10 units were placed in a 4000×4000 continuous-space arena with periodic boundaries (i.e., a torus) featuring a centrally located circular base with a diameter of 100 units. One simulated second consisted of 50 update loops (hereafter "updates") and an experimental run lasted 600 simulated seconds. One robot travelling at top speed in a straight line could traverse the full length of the world in 80 seconds (i.e., travelling at 50 units per second).

A set of foraging scenarios were defined, each characterised by the number, N_D , of 5×5 resource deposits in the environment and their properties. Each deposit, i, comprised V_i units of material, where each unit had a deposit-specific resource quality, $Q_i \in [0,1]$, such that the total resource in a deposit was equal to V_iQ_i . Each robot could extract up to one unit of volume from a resource deposit per visit, reducing the deposit's volume and extracting Q_i resource to be returned to the robot base. Once the entire volume of a resource deposit had been collected it ceased to exist in the environment and was not replaced.

By default, all deposits were distributed uniformly at random in space with each deposit having equivalent quality (Q=0.5). However, some environments featured non-uniform spatial distributions comprising a number of spatially clustered groups of deposits, and, additionally, some environments featured non-uniform quality distributions in the form of a number of higher-quality patches. Note that these patches did not influence the spatial distribution of deposits, only their quality.

Where the number of spatially clustered groups, N_G , was greater than zero, each of these groups featured 10 deposits

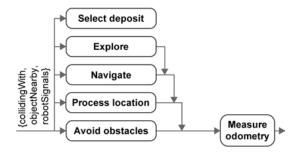


Figure 1: Subsumption architecture of the robot controller.

clustered within a 500-unit radius circular region and no deposits existed outside these groups. Where the number of quality patches, N_P , was greater than zero, each patch was centered on a deposit which was allocated Q=1, and the quality of the deposits around it decayed with distance, d, from the patch centre:

$$Q = \min\left(e^{-d^2/\beta\sigma_P^2}, \ 0.5\right),\tag{1}$$

where patch quality variance, σ_P^2 , was a scenario-specific parameter that controlled the size of the quality patch, and β was a constant factor set to 8×10^6 for all results reported here. In such scenarios, the quality, Q, of deposits outside of patches was scaled down by a constant factor in order to keep the total amount of net resource equal across scenarios. Quality patch centres were randomly chosen deposits with no more than one patch centre per group (when $N_G > 0$).

Robots

The simulated robots utilised a subsumption architecture (Brooks, 1986), where low-level behavioural modules such as avoiding obstacles could be overriden by motor commands of higher-level behaviours such as navigation towards a target (Figure 1). The individual modules communicated with each other by setting the robot's state and other internal variables stored in the robot's memory. A finite-state machine representation of the robot is depicted in Figure 2.

Robots were initially randomly oriented and located within the central home base which featured a beacon that allowed any robot to navigate directly home from any point in the arena (Pini et al., 2013).

Individualist robots (I-Swarm) performed a random walk to search for food, avoiding each other by backing up and turning by a random angle. When a deposit was discovered, one unit of volume was loaded (or the entire deposit if one or less than one unit remained) and returned to the base (navigating using the home beacon). On the basis of subjective odometry calculations, a robot returned to where it estimated the deposit was located if the energy efficiency, E_e , of the deposit was better than that of the worst deposit that it had found so far.

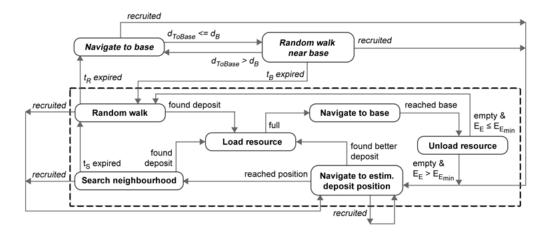


Figure 2: Finite-state machine representation of the robot controller. B-swarm logic is shown in italics outside the dashed box.

Estimation of energy efficiency was inspired by a model of bees (Seeley, 1994) and took into account the volume, V, and quality, Q, of the deposit, as well as the cost of revisiting it, calculated in terms of time and energy required to (i) reach it (1 unit of energy per simulation update), (ii) load it (5 seconds at 5 units of energy per update), (iii) return to base (5 units of energy per update) and (iv) unload it (2 seconds at 5 units per update):

$$E_e \sim \mathcal{N}\left(\frac{VQ-C}{C}, \sigma_e^2\right)$$
 (2)

$$C = \sum_{i=1}^{4} T_i * E_i, \tag{3}$$

where $\sigma_e^2 = 0.05$ was the variance of normally distributed estimation noise, and T_i and E_i were the time and energy per unit time required for each of the four foraging subtasks. It was also assumed that a robot could estimate the energy efficiency of a deposit from a maximum distance of 30 units.

Upon reaching a remembered location, neighbourhood search was undertaken for a maximum of $t_S=10$ seconds, comprising a random walk constrained within a distance of $r_S=120$ units around the estimated deposit location (Pini et al., 2013). The robot resumed random search if nothing was found, or if it could not reach the remembered location within 120 seconds. If, en route to a remembered location, a robot encountered a new deposit with superior energy efficiency, the superior deposit would be visited instead.

Odometry was used to estimate the location of a remembered deposit based on a robot's subjective impression of its rotational change $\Delta \alpha'$ and speed S' during locomotion. Odometry was affected by Gaussian noise on the actual values of rotation (α) and speed (S), simulating wheel slippage,

etc. (Chong and Kleeman, 1997; Martinelli, 2002):

$$\Delta \alpha' \sim \mathcal{N} \left(\Delta \alpha, \ \kappa \times \Delta \alpha \right),$$

 $S' \sim \mathcal{N} \left(S, \ \kappa \times S \right),$
(4)

where parameter κ scaled the variance of odometry noise.

Collective foragers (B-Swarm) employed the same basic foraging strategy as the I-Swarm, but could also signal to nearby robots (< 60 units) their estimate of a remembered deposit's location and its associated energy efficiency (Jevtic and Gazi, 2010). A robot was recruited to a signalled location if it either did not have the location of a deposit in memory, or if its remembered deposit had inferior energy efficiency. Additionally, a B-Swarm robot returned to the base and roamed within $d_B=200$ units of its centre for up to $t_B=120$ seconds if it could not find a deposit during $t_R=120$ seconds of random walk. In this way, unsuccessful foragers could discover deposit locations from robots returning to base to unload, mimicking to some extent the behaviour of bees (Biesmeijer and de Vries, 2001; Granovskiy et al., 2012).

Results

Three sets of simulation results are reported here. First, we explore the impact of varying both the number of robots and the number of deposits on I-Swarm performance in order to identify values for N_R and N_D that present an appropriate degree of challenge. Second, we compare the performance of I-Swarms and B-swarms over a range of foraging scenarios that differ in key respects. Third we explore the impact of odometry error on swarm foraging performance.

Calibrating N_R and N_D

I-Swarm performance was assessed in two basic scenarios, *Uniform* quality and *Patchy* quality, varying the number of

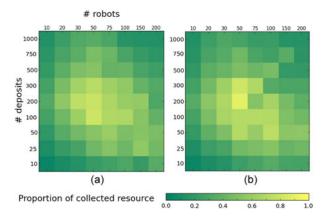


Figure 3: I-Swarm performance as a function of the number of robots and number of deposits in (a) *Uniform* quality and (b) *Patchy* quality scenarios. Each data point is the mean of 10 independent simulations.

robots, N_R , and the number of deposits, N_D (see Figure 3). In both scenarios, $V=200/N_D$, and deposits were distributed uniformly at random across the entire arena. *Uniform* quality deposits had Q=0.5. The quality of *Patchy* deposits was influenced by $N_P=30$ randomly located quality patches, each with $\sigma_P^2=0.01$.

In both scenarios, performance peaked for an intermediate number of robots and an intermediate number of deposits. With too few robots or too many deposits, not enough foraging trips could be carried out within the duration of a simulation. With too many robots performance deteriorated due to physical interference between foragers near the base. With too few deposits, performance was reduced by the increased challenge involved in discovering deposits far from the base.

In the *Patchy* quality scenario there was some evidence that performance depended on an interaction between N_D and N_R (influencing the degree of interference between robots) but the peak and overall shape of the performance distribution was not significantly altered. Consequently, in order to assess swarms in scenarios that varied in the challenge that they presented, and thereby avoid floor and ceiling effects, a range of values for $N_D \in [10,100]$ and $N_R \in [10\dots 100]$ were established and utilised for the remainder of the results presented below.

I-Swarm and B-Swarm Foraging Performance

A set of foraging scenarios were defined, varying in spatial distribution, volume, quality, etc.:

- Litter scattered in a field: a uniform random spatial distribution of equal quality deposits, with $N_D = 100, N_G = 0, N_P = 0, V = 2, Q = 0.5$.
- Large **Puddles** of rain water: like *Litter*, except deposits are ten times larger, $N_D = 10, V = 20$.

- Rare **Minerals**: a uniform random spatial distribution of patchy quality deposits located located at least 1500 units from the base, with $N_D=10, N_G=0, N_P=5, \sigma_P^2=0, V=20, Q=$ varied.
- Common Stones: like *Minerals*, but $N_D = 100, N_P = 10, \sigma_P^2 = 0.01, V = 2, Q = \text{varied}.$
- Nectar from flower clusters: 10 groups of 10 deposits each. Location of group centres was drawn from a uniform random distribution at least 500 units away from the base, with $N_D=100, N_G=10, N_P=3, \sigma_P^2=0.01, V=2, Q=$ varied.
- Lost **Cargo**: like *Nectar*, but with one group of equal quality deposits where $N_D=10,N_G=1,N_P=0,\sigma_P^2=0,V=20,Q=0.5$.

Variation in swarm performance across different scenarios (see Figure 4) can be explained in terms of three types of robot-robot intereference:

- 1. Physical interference caused by one robot obstructing another typically while foraging from the same deposit or returning to the base at the same time.
- Environmental intereference where one robot substantively alters the foraging environment of another by depleting its current target deposit.
- Informational interference, unique to the B-Swarm, where one robot's behaviour is influenced either positively or negatively by signals from another robot.

I-Swarms slightly but consistently outperformed B-Swarms when collecting Litter in swarms of less than 100 This advantage was statistically significant for swarms of size $N_R = 10$ and 20, and also significant across all $N_R < 100$ swarms when their results were considered together (Wilcoxon signed-rank test, p < 0.01). The fact that litter deposits were abundant and could be depleted in two visits meant B-Swarms suffered from informational interference that negatively impacted on performance. In theory, Stones presented a more significant challenge for both swarms since they were located further from the base and were, at least initially, harder to find. However, the slight advantage that I-Swarms experienced for Litter was extinguished in this scenario since B-Swarms could benefit from recruiting robots to the general location of the stones, and were thus able to match I-Swarm performance.

B-Swarms were able to collect more resources than I-Swarms in each of the other scenarios, where deposits were richer in volume but more difficult to discover, i.e., *Puddles*, *Minerals*, *Nectar* and, most strikingly, *Cargo* where all deposits were in one area far from the base. In this scenario, a B-Swarm of 100 robots was even able to collect more resource than an I-Swarm of the same size in an easier, *Nectar* collection task.

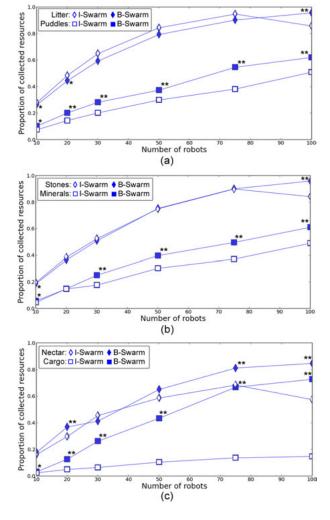


Figure 4: The influence of scenario type on the foraging performance (measured as the median proportion of resource collected over 20 independent simulation runs) of I-Swarms and B-Swarms of different sizes. Statistically significant differences between I-Swarm and B-Swarm performance in the same scenario are indicated with asterisks (Wilcoxon signed-rank test, ** p < 0.01, * p < 0.05). The average interquartile range was approximately 0.082.

An interesting effect of recruitment was also observed when 100-robot swarms foraged for *Litter*, *Stones* or *Nectar*. I-Swarms of this size suffered a drop in performance resulting from physical interference in the form of congestion around the base. By contrast, recruitment in B-Swarms caused smaller groups of robots to forage from neighbouring locations, meaning that foragers from one group solved the problem of avoiding each other near deposits in parallel with other groups. Such self-organisation led to a kind of spontaneous traffic management and consequent reduction

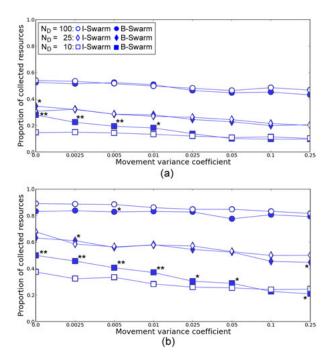


Figure 5: The influence of odometry error (κ) on the performance (measured as the median proportion of resource collected over 20 independent simulation runs) of I-Swarms and B-Swarms of different sizes in the *Variable* scenario: (a) $N_R=20$, (b) $N_R=50$. Statistically significant differences between I-Swarm and B-Swarm performance in the same scenario are indicated with asterisks (Wilcoxon signed-rank test, ** p<0.01, * p<0.05). The average interquartile range was approximately 0.072.

of near-base congestion.

On the other hand, there were cases where recruitment did not improve foraging performance. Mineral desposits, for instance, were very hard to find for small swarms. B-Swarms needed to have at least 30 members before it was likely that they could find a deposit early enough to ensure that recruitment enabled them to outperform an I-Swarm. Moreover, while B-Swarms typically outperformed I-Swarms when foraging for Nectar, this did not hold for swarms of size $N_R=30$ and 50. Under these conditions, B-Swarms were not large enough to discover and exploit multiple groups of flowers in parallel, but were large enough to rapidly exhaust a single group of flowers by recruitment, ensuring that B-Swarms spent significant time returning to exhausted groups, and proportionately more time randomly searching rather than benefitting from recruitment. This unfortunate interplay between rapid exploitation of a group and insufficient exploration caused B-Swarms of 30 and 50 robots to perform similarly to I-Swarms of the same size.

Odometry error

Error in a robot's position estimation is an important influence on foraging performance where robots return to remembered deposits or are recruited to new ones. In this section, the value of the κ parameter (Equation 4), previously set to 0.005, is varied in order to explore the effect of odometry noise in the *Patchy* scenario (see Figure 5).

Confirming the previously presented results with low odometry noise, the B-Swarm outperformed the I-Swarm when $N_D=10$, and performed similarly or slightly worse than the I-Swarm when the number of deposits increased. Increasing κ did not affect the performance of either swarm when deposits were very easy to find $(N_D=100)$ and only slightly degraded the performance when $N_D=25$.

The impact of odometry error was most significant when 10 deposits were placed in the environment. While collective foragers could harvest significantly more resources than individualists when odometry error was low, the relative advantage of B-Swarms degraded as κ increased until it was extinguished or even slightly revered. This was an expected result, as the information about location of deposits that guided the B-Swarm robots became more erroneous with high κ , while individualist errors did not propagate to other robots in the I-Swarm, the performance of which was generally less affected by κ .

Interestingly, the impact of odometry noise on B-Swarm performance was stronger for 50-robot swarms than 20-robot swarms. While the performance of a 20-robot B-Swarm rarely fell below that of an equivalent I-Swarm even for high levels of odometry noise, 50-robot B-Swarms tended to perform worse than equivalent I-Swarms for moderate or high levels of odometry noise.

Even though robots in a 50-robot B-Swarm tended to recruit each other at a similar rate, the difficulty of finding advertised deposits when κ was high resulted in increased recruitment to deposits of low quality. For such swarms, the proportion of deposits within a higher-quality patch visited due to recruitment was 21% when $\kappa=0.01$, but only 13% when $\kappa=0.1$. Moreover, the distortion of information due to higher odometry error resulted in a smaller total amount of deposit collections (average of 76.9 visits per run when $\kappa=0.01$, but only 47.1 when $\kappa=0.1$).

In contrast, while increased odometry error also negatively affected the amount of times deposits were visited by a B-Swarm of 20 robots (average of 32.5 times per run when κ =0.01; 20.6 times when κ =0.1), the smaller swarm was able to switch to more individualist behaviour when robots failed to locate advertised deposits due to poor odometry: 52% of visited deposits were found by random walk and 33% by recruitment when κ =0.01, while 63% were found by random walk and only 23% by recruitment when κ =0.1.

On the other hand, in the larger 50-robot B-Swarm it was harder to behave like an individualist when odometry failed (34% of deposit visits were due to recruitment when $\kappa =$

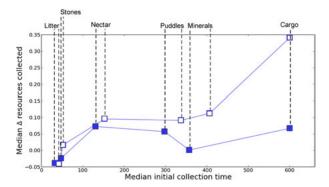


Figure 6: The relationship between the median time taken to collect the first resource and the difference between total resource collected by a B-Swarm and an I-Swarm in various scenarios. Data is shown for 20-robot swarms (solid symbols) and 50-robot swarms (open symbols).

0.01, with a very similar 31% due to recruitment when $\kappa = 0.1$) since a bigger group generated more ultimately fruitless recruitment. If at least one searching robot eventually found a deposit, this information propagated through the swarm, causing the robots to remain near the location and try to access the resource, generating wasteful congestion instead of allowing the resumption of potentially more useful random walk exploration.

Discussion

It is often difficult to intuitively predict the aggregate results or systemic properties of swarm behaviour because of the non-trivial nature of inter-robot interactions and interactions between robots and their environment. However, the analysis presented in this paper shows that it is possible to understand swarms by studying how they behave in various environments. Here we offer a set of generalised principles extracted from the experiments.

When To Forage Collectively

1. When resource is hard to find: B-Swarms performed better than I-Swarms in environments where deposits were harder to find but returned more resource. In particular, the advantage of recruitment was related to the difficulty of discovering deposits (Figure 6), but not to the length of a trip from a deposit to the base, at least within the scale of the scenarios simulated here. The B-Swarm advantage was more significant in larger swarms that covered the area faster and thus had a higher probability of discovering deposits. A small B-Swarm of 20 robots had difficulties, especially where deposit discoveries were rare and represented a small fraction of the total resource. These results suggest that communicating robots are useful for tasks like extracting rare minerals or retrieving lost cargo, but not for picking up litter from streets.

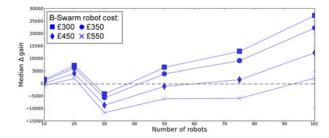


Figure 7: Projected relative gain from using B-Swarm rather than I-Swarm to collect *Nectar*, where the cost of an I-Swarm robot is £300, the gain from collecting all resources is £100,000, and the cost of a B-Swarm robot varies.

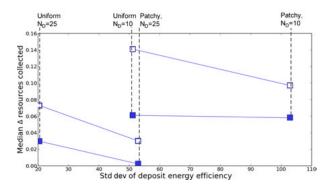


Figure 8: The relationship between the standard deviation of deposit energy efficiency and the performance difference between B-Swarms and I-Swarms. Data is shown for 20-robot swarms (solid symbols) and 50-robot swarms (open symbols). The B-Swarm advantage did not increase in the *Patchy* variant of each *Uniform* scenario.

2. When congestion is a problem: The performance of large swarms of individualists decreased rapidly as congestion around the base prevented robots from foraging. In contrast, recruited groups of B-Swarm robots could solve congestion problems near deposits in parallel with each other, increasing the amount of robots that could operate effectively near the base at the same time. It is notable that this B-Swarm feature would become less effective if the swarm size increased significantly, as the recruited groups would become larger and there would be more of them.

When Not To Forage Collectively

1. When resource is abundant: I-Swarms could slightly outperform B-Swarms when items were abundant and easy to find. Not only was recruitment in these environments unnecessary, but a combination of environmental and informational interference caused advertised locations to become depleted before a recruit could reach them, ensuring that recruitment incurred a performance cost.

- **2.** When reliablity of information is low: A decrease in the reliability of information about deposit locations due to odometry error caused the foraging ability of collective foragers to degrade significantly. In these cases, I-Swarm performance may be more reliable, although still generally poor.
- **3.** When deposits are very hard to find quickly: It was very difficult for small swarms of both types to find *Minerals* and *Cargo* where deposits were concentrated in a few distant locations. Despite the additional capabilities of the B-Swarm, once deposits were discovered there was not enough time to recruit others and exploit the location information.
- **4. Borderline cases:** In the real world, the cost of a robot swarm must be reasonable when assessed against the value generated by the foraging task. The additional investment required in order to equip robots with communication transmitters and receivers, and program them with the extended B-Swarm logic should thus be taken into account. For example, imagine that an I-Swarm robot costs £300 and that collecting all resources from the environment is worth £100,000. If the additional cost of a B-Swarm robot is £50, the effect of such an investment is not very dramatic (Figure 7). However, as communicating robots become more expensive, they are less able to return the extra investment.

Further Remarks

As currently implemented, B-Swarms were not better equipped than I-Swarms to selectively forage from more energy efficient deposits (Figure 8). Since B-Swarm robots were recruited to any deposit with an energy efficiency higher than the lowest known E_e , a large proportion of the swarm would collect from mediocre locations instead of exploring the environment. However, it was observed that unemployed B-Swarm robots could target higher E_e deposits as a consequence of receiving information from a large number of returning foragers. It is thus possible that the ability to preferentially exploit deposits with high E_e could be achieved by giving the robots a memory of previously encountered resources and an adaptive E_e threshold that would filter social information, as is the case in bees (Seeley, 1994). It would also be possible to implement a region of the robot base that acts as a dance floor in a beehive (Seeley, 1994; Wray et al., 2011), where robots would meet and compare advertised deposits in a more centralised fashion.

Robots designated for exploration would potentially also improve the performance of the swarm. It was shown that active scouting followed by rapid recruitment are very important factors in making bees so successful in dynamic environments (Granovskiy et al., 2012), compared to other social insects like ants (Ribeiro et al., 2009). Individual effects of the bee-inspired behaviours need to be explored in future experiments in order to establish their importance.

Conclusion

The ability of a simulated robotic swarm of individualist foragers and a swarm of collective foragers to harvect resources from various environments was compared. It was shown that recruitment is useful when resources are hard to find and are either clustered spatially or reward multiple visits, as the robots can spend relatively more time exploiting known deposits than exploring for new ones. On the other hand, the additional cost associated with building communicating robots may not be justified when there are many locations from which to gather resource or when the return expected from repeated visits to deposits is low. Furthermore, robots that rely on communication are more susceptible to errors when positional information looses reliability as may occur through odometry error.

Despite the tempting intuition that better or more expensive swarms will deliver better results, the question of whether to equip robots with the ability to communicate is not a trivial one. Just as natural evolution shapes species to become as simple but at the same time as effective as possible within their biological niche, engineered robots can benefit from our ability to step back and consider the nature of the tasks that they face.

Acknowledgements

This work was supported by an EPSRC Doctoral Training Centre grant (EP/G03690X/1).

Source code: http://lenkaspace.net/lab/swarmSystems/alife14

References

- Arab, A., Carollo Blanco, Y., and Costa-Leonardo, A. M. (2012). Dynamics of foraging and recruitment behavior in the Asian subterranean termite *Coptotermes gestroi* (Rhinotermitidae). *Psyche*, 2012. doi:10.1155/2012/806782.
- Beekman, M. (2001). Phase transition between disordered and ordered foraging in Pharaohs ants. *P. Natl. Acad. Sci. USA*, 98(17):9703–9706.
- Biesmeijer, J. C. and de Vries, H. (2001). Exploration and exploitation of food sources by social insect colonies: A revision of the scout-recruit concept. *Behav. Ecol. Sociobiol.*, 49(2):89–99.
- Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE J. Robotic. Autom.*, 2(1):114–23.
- Busse, C. D. (1978). Do chimpanzees hunt cooperatively? *Am. Nat.*, 112(986):767–770.
- Chong, K. S. and Kleeman, L. (1997). Accurate odometry and error modelling for a mobile robot. In *IEEE Int. Conf. Robotics & Automation*, pages 2783–2788.
- Gittleman, J. L. (1989). Carnivore group living: Comparative trends. In Gittleman, J. L., editor, *Carnivore Behavior, Ecology and Evolution*, pages 183–207. Cornell University Press.

- Granovskiy, B., Latty, T., Duncan, M., Sumpter, D. J. T., and Beekman, M. (2012). How dancing honey bees keep track of changes: The role of inspector bees. *Behav. Ecol.*, 23(3):588–596.
- Hoff, N., Sagoff, A., Wood, R. J., and Nagpal, R. (2010). Two foraging algorithms for robot swarms using only local communication. In *IEEE Int. Conf. Robotics & Biomimetics, ROBIO* 2010, pages 123–130. IEEE Press.
- Holekamp, K. E., Smith, J. E., Strelioff, C. C., Van Horn, R. C., and Watts, H. E. (2012). Society, demography and genetic structure in the spotted hyena. *Mol. Ecol.*, 21(3):613–32.
- Jevtic, A. and Gazi, P. (2010). Building a swarm of robotic bees. In World Automation Congress, pages 1–6. IEEE Press.
- Lein, A. and Vaughan, R. T. (2009). Adapting to non-uniform resource distributions in robotic swarm foraging through worksite relocation. In *IEEE/RSJ Int. Conf. Intelligent Robots & Systems*, pages 601–606. IEEE Press.
- Martinelli, A. (2002). The odometry error of a mobile robot with a synchronous drive system. *IEEE T. Robotic. Autom.*, 18(3):399–405.
- Pini, G., Brutschy, A., Pinciroli, C., Dorigo, M., and Birattari, M. (2013). Autonomous task partitioning in robot foraging: An approach based on cost estimation. *Adapt. Behav.*, 21(2):118– 136
- Ribeiro, P. L., Helene, A. F., Xavier, G., Navas, C., and Ribeiro, F. L. (2009). Ants can learn to forage on one-way trails. *PLoS One*, 4(4):e5024.
- Robinson, S. K. and Holmes, R. T. (1982). Foraging behaviour of forest birds: The relationships among search tactics, diet and habitat structure. *Ecology*, 63(6):1918–1931.
- Rybski, P. E., Larson, A., Veeraraghavan, H., Lapoint, M., and Gini, M. (2004). Communication strategies in multi-robot search and retrieval: Experiences with MinDART. In Alami, R., editor, *Proc. 7th Int. Symp. Dist. Auton. Robotic Systems*, pages 301–310.
- Sarker, M. O. F. and Dahl, T. (2011). Bio-inspired communication for self-regulated multi-robot systems. In Yasuda, T., editor, *Multi-Robot Systems, Trends and Development*, pages 367–392. InTech.
- Seeley, T. D. (1992). The tremble dance of the honey bee: Message and meanings. *Behav. Ecol. Sociobiol.*, 31(6):375–383.
- Seeley, T. D. (1994). Honey bee foragers as sensory units of their colonies. *Behav. Ecol. Sociobiol.*, 34(1):51–62.
- Shell, D. A. and Mataric, M. J. (2006). On foraging strategies for large-scale multi-robot systems. In *IEEE/RSJ Int. Conf. Intelligent Robots & Systems*, pages 2717 – 2723. IEEE Press.
- Stander, P. E. and Albon, S. D. (1993). Hunting success of lions in a semi-arid environment. *Symp. Zool. S.*, 65:127–143.
- Weimerskirch, H., Le Corre, M., Jaquemet, S., Potier, M., and Marsac, F. (2004). Foraging strategy of a top predator in tropical waters: Great frigatebirds in the Mozambique Channel. *Marine Ecology Progress Series*, 275:297–308.
- Wray, M. K., Klein, B. A., and Seeley, T. D. (2011). Honey bees use social information in waggle dances more fully when foraging errors are more costly. *Behav. Ecol.*, 23(1):125–131.

Embodied Evolutionary Robotics with Large Number of Robots

Nicolas Bredeche^{1,2}

¹Sorbonne Universités, UPMC Univ Paris 06, UMR 7222, ISIR, F-75005, Paris, France ²CNRS, UMR 7222, ISIR, F-75005, Paris, France nicolas.bredeche@upmc.fr

Abstract

Embodied evolutionary robotics is a particular flavour of evolutionary robotics, where the evolutionary optimization of behaviours is achieved in an on-line and distributed fashion (Watson et al., 2002). The question asked in this paper is: does population size play a role in the evolution of particular behaviours? We experimentally demonstrate that varying the number of robots and the size of the environment can lead to very different outcomes in terms of evolved behaviours.

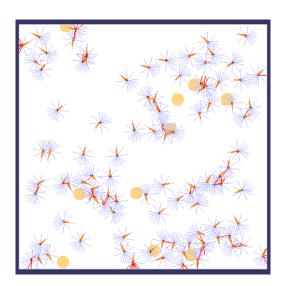


Figure 1: Experimental setup: a population of robots with 8 infra-red (IR) sensors (shown in blue) is deployed in an environment where 10 (yellow) landmarks are randomly placed. The robots are modelled after the famous e-puck robot, and communication between robots is achieved through the IR devices. The red tail is visible to the user only (used for identifying directions).

The mEDEA algorithm

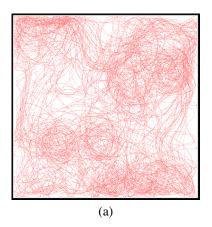
For this study, we use a particular algorithm that does not address any objective function, but rather perform environment-driven evolution. Similar to other works in artificial life, such as Tierra (Ray, 1991), medea favors individuals that are capable to spread their genomes in other individuals, without any explicit consideration for any user-defined task. medea has been extensively described elsewhere (Bredeche et al., 2012), and has been tested with up to 20 real robots.

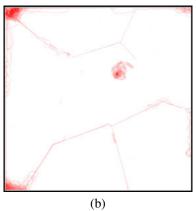
medea considers a set of robots with limited communication capabilities. Evolution relies on the diffusion of genomes from peer to peer, and genomes compete with one another to spread over the population of robots. To do so, each robot carries a genome, which defines its behaviour during a predefined amount of time (its "lifetime"). Whenever two robots are close enough, each transmits a mutated copy of its current genome to the other, and store the incoming genome in a list for further use. At the end of the robot's lifetime, the list of previously received genomes is emptied except for one arbitrarily selected genome, which is then mutated (using gaussian mutation) and replaces the genome used so far. Because the new genome is arbitrarily selected, there is pressure towards genomes that are able to drive their "vehicles" (i.e., the robots) to spread themselves over the population, favouring behavioural strategies better fitted to the environment.

Experiments

In a previous work (Bredeche et al., 2012), we showed that the average number of encounters is critical to survival, and is related to the number of robots, the communication radius and the size of the environment. That is: if robots get lost and end up with no new genomes to carry on evolution, the algorithm fails (ie. the robots do not survive).

The question addressed here is quite the opposite: given the population is large enough for the algorithm to work (ie. each robot will encounter at least one other robot during its lifetime), what happens when we consider larger populations? To answer this question, we consider experiments with different setups (units are given in pixels, a robot is 5x5 pixels): (1) Setup 1: 100 robots in a 400x400 environment; (2) Setup 2: 200 robots in a 1000x1000 environment; (3)





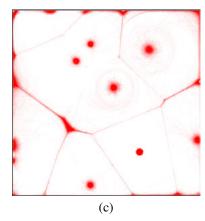


Figure 2: Trajectories from the last generations of three different setups (typical runs): (a) 100 robots, 400 x 400 environment; (b) 200 robots, 1000 x 1000 environments; (c) 4000 robots, 4000 x 4000 robots.

Setup 3: 4000 robots in a 4000x4000 environment.

The setup we consider is similar to what is shown in Figure 1 (see caption for description), except for the environment size and number of robots which vary wrt. the setups. We consider that each robot lives for 400 steps, during which it spreads its genome to every robots encountered. Each robot senses its surrounding with 12 infra-red sensors with limited range, which are also used for local communication (sending/receiving genomes), as well as a particular sensor which gives the direction and distance to the closest landmark (infinite range). There are 10 landmarks in the environment (initially placed at random locations). Landmarks provide no direct advantage nor disadvantage to nearby robots (ie. landmarks are a priori useless). Each robot is controlled by an Elman recurrent neural network with 5 neurons in the hidden layer, and the network weights are read from the genome.

Fig. 2 shows three typical examples from each of the three setups (at least 10 replications per setup have been done, each leading to similar results). These images show the trajectories of each of the robots over the last few generations of one run. Several conclusions can be drawn just from experimental observations.

Setup 1 displays randomly moving robots. This is an efficient behavioural strategy as each robot gets the opportunity to spread its own genome, while not being able to distinguish itself from others by doing so. Though a tendency to orbit around landmarks can be seen, this is far from the only behaviours observed, and behaviours from all robots are very similar with no clear clusters identified;

Setup 2 is rather different as three different behaviours can be observed: (1) some robots gather in corners (2) a group of robots is closely orbiting around one of the 10 landmarks and (3) some (few) robots appears to be travelling in straight lines.

Setup 3 provides a clear vision of what Setup 2 merely sketched: robots are either orbiting around landmarks, or

travelling in straight lines away from the landmarks. Also, very small groups are found orbiting around landmarks, but at some distance, avoiding interaction with the robots which are positioned right on the landmark (e.g. the landmark in the center).

These experiments show that increasing the number of robots and size of the environment leads to more diversity in the behaviours observed. However, the most compelling observation comes from robots travelling in straight lines, seemingly avoiding the landmarks. While it was not predicted *a priori*, this behaviour can be understood from observations: robots keep the farther away from any landmarks, resulting in clusters of robots travelling on the exact frontier between two landmarks. It is also interesting, if not useful, to note that this type of behaviour produces some kind of distributed voronoi tesselation, using landmarks as seeds.

Conclusion

The take-home message is that population size may dramatically change the results of evolution. In particular, the emergence of what resembles distributed Voronoi tesselation shows a striking example of what can only be observed when scaling up the population size.

One current limitation of this study it that population size and environment size were changed simultaneously, without considering the *density* of robots. This is currently under investigation.

Acknowledgements

The author wishes to thank Leo Cazenille, Arthur Bernard and Jean-Baptiste Mouret for comments.

References

Bredeche, N., Montanier, J.-M., Wenguo, L., and Winfield, A. F. (2012). Environment-driven Distributed Evolutionary Adaptation in a Population of Autonomous Robotic Agents. *Mathematical and Computer Modelling of Dynamical Systems*, 18(1).

Ray, T. (1991). Is it alive, or is it ga? In Proceedings of International Conference on Genetic Algorithms.

Watson, R. A., Ficici, S. G., and Pollack, J. B. (2002). Embodied evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, 39(1):1–18.

Associating Nearby Robots to their Voices

Plamen Ivanov and Dylan Shell Texas A&M University, College Station, TX 77840 flame09@tamu.edu

Abstract

It can be useful for a robot in a team to recruit the help of others after considering their poses in space. When the initiating robot is aware of the network addresses (or IDs) of the robots it may ask for cooperation over the network. But if the initiating robot is ignorant of their IDs then it becomes rather more complex or time consuming. In this paper, robots use a simple visual (locally sensed) gesture (such as a light being turned on or off) in addition to wireless messages to quickly and effectively establish a one-to-one association between physical location IDs and wireless IDs. This paper identifies, describes, formalizes, and then generalizes the problem of establishing a local association between neighboring robots' physical location IDs (in the reference frame of an observing robot) and their wireless IDs. We describe three algorithms that solve this problem: two are deterministic algorithms and one is a probabilistic algorithm. Using the algorithms and formalism, we examine the structure underlying this association problem.

Introduction

Robots cooperating in teams often benefit from spatially referenced communication: "Would the robot to my left, please help me move this piano?" But many current implementations are influenced by the fact that practical perception of other robots is sensor-based (e.g., using vision via cameras) while communication is based on network technology (e.g., Wi-Fi, radios). Since the same robot will have different identifiers in these two realms, an association must be formed if our robot is to request help over the network: "Will robot 192.168.0.56 help move this piano?" We consider the problem of associating two independent channels of communication to form a one-to-one matching between sources in one channel and communication sources in the other.

We specifically consider the question of relating a robot with some relative pose (*i.e.*, when the "where" of the robot is known in the local frame) to a name which can be used to refer to that robot directly (*i.e.*, the "who" of that robot). The robots comprising the system are assumed to be able to interact through two independent communication channels. Using only these two channels and no global localization knowledge or global communication, all robots in the system must each solve:

For each robot "visible" in the second channel, determine and associate the appropriate network address in the first channel.

The many different variants of this problem are usually addressed by using an array of visual fiducials and a hand-crafted list of associations (Garrido-Jurado et al., 2014; Cassinis and Tampalini, 2007; Howard et al., 2004), or by initially calibrating and identifying the robots. These solutions are *ad hoc* and focus on practical cases of a specific variant of the problem (Mathews et al., 2010).

In terms defined by Støy (2001), the first channel in the motivating system is a non-situated channel, such as a Wi-Fi or radio, and the second channel is a situated communication channel, such as a light source attached to a robot. The capabilities of the situated visual channel are assumed to be minimal: a simple on/off communication strategy over the visual channel is enough to complete the association successfully. This could be implemented by moving the robot (e.g., performing a predefined wiggling repertoire), or switching on a light, or other such visually identifiable gesture (e.g., an instance of a protocol on just such a channel is given in Dieudonné et al. (2009)). The non-situated radio channel is assumed to have a message size which can at minimum fit the radio ID of a robot.

In this paper, we identify and formalize the problem generally, showing that it has an interpretation from an information filtering perspective (LaValle, 2009). Our paper describes two deterministic algorithms and a probabilistic algorithm solving this association problem, each having distinct features and drawbacks. The formalism and solutions are applicable to many circumstances in which indexical knowledge must be related to objective knowledge (Agre and Chapman, 1987; Lespérance and Levesque, 1994) because the former represents task relevant information—natural in the one channel—while the latter permits targeted transmission—a straight-forward operation in the other channel.

The generalized association problem impinges on questions of task representation, addressing, communication, and localization —each topics that are subtle and nuanced in their own right. For example, although in the motivating

piano moving scenario the indexical, task-relevant channel is the visual, situated channel, this need not be the case. Using this formalism we have made a step towards identifying and understanding the subtleties involved.

Related Work

Practically speaking, marker systems are limited in the number of unique identifiers which can be encoded, suffer from reliability issues of correctly identifying markers, have restrictions on the maximum distance at which markers can be identified and in what environments they can be deployed, and have significant hardware and software requirements (Garrido-Jurado et al., 2014; Cassinis and Tampalini, 2007). Solutions that require calibration of global knowledge are especially undesirable as system size increases.

The idea of *spatially targeted communication* in Mathews et al. (2010) and the model and solutions they use have similarities with the model and solutions we describe. However, their approach is only a solution for a pair of robots in the system. The model lends itself to analysis through the extinction probability of a branching processes (Haccou et al., 2007); we return to discuss this means of analysis later. Dieudonné et al. (2009) present a multi-robot system in which robots are indistinguishable from each other and there is no explicit means of communication. Their approach is remarkable because it considers a weak synchronicity model, but it makes strong sensing assumptions, precluding use in most real robot settings.

Mutual or collective localization in a multi-robot system (e.g., Franchi et al. (2009); Fox et al. (2000)) is related to the association problem presented in this paper because a global metric reference frame provides unique labels for each robot. Oftentimes, however, the cost of localizing all the robots is greater than one wishes to pay merely for targeted communication.

Problem Definition and Formalism

In this section we give a precise description of the multirobot system and the problem which we address.

Motivating System

Consider a multi-robot system composed of n robots, which are spread arbitrarily across an environment. Each robot in the system can directly observe and distinguish from the environment some nearby robots using a camera. Robots can also flash a light on and off. The camera and light form a single bit communication channel — the **camera channel**. Each robot can also directly communicate through messages over a **wireless channel** with robots in range. Each robot has a **wireless ID** and itself assigns **location IDs** (for example, local range and bearing pairs) to robots it can recognize through its camera.

Every robot starts knowing n and the IDs of robots they can receive messages from in either of the channels. The robots have no knowledge of which wireless ID is assigned

to which robot. The operation of the robots is modeled with synchronous, discrete time steps. Both channels have a limited transmission range *i.e.*, a robot is unable to receive messages from all robots in the system directly.

The goal is for each robot to form a one-to-one association between the location IDs of robots in the camera channel and the wireless IDs of robots in the wireless channel.

Generalized Formal Specification

A generalized formal definition of the problem follows. Graphs representing the system are depicted in Fig. 1.

Definition 1. We model a multi-robot system that has one radio communication channel (channel 1) and one physically situated communication channel (channel 2) with the tuple: $\langle G_1 = \langle V, E_1 \rangle, G_2 = \langle V, E_2 \rangle, C_1, C_2, f_1, f_2 \rangle$ where

- 1. V is a set of vertices, each representing a robot.
- 2. $G_1 = \langle V, E_1 \rangle$ and $G_2 = \langle V, E_2 \rangle$ are two directed graphs representing the connectivity of the system of robots over the two communication channels,
- 3. $E_1 \subseteq V \times V$ has directed edges s.t. $e \equiv (v_i, v_j) \in E_1$. This is interpreted as robot i being able to receive a one-hop message from robot j over channel 1,
- 4. $E_2 \subseteq V \times V$ is analogous, but for channel 2,
- 5. $E_2 \subseteq E_1$ this is the channel **inclusion property**, stating that if i can receive a channel 2 message from a robot then it is guaranteed to be able to receive a channel 1 message from that robot too,
- 6. f_1 , f_2 are labeling functions of the form $f: V \times V \mapsto C$ which are applied to channels 1 and 2, respectively.
- 7. C_1 and C_2 are the label sets for channels 1 and 2.

Each channel has an associated labeling function and the set of labels correspond to addresses (wireless IDs and location IDs, respectively). In the motivating system, the labeling functions will provide unique IDs for the wireless channel (IP addressed) and locally unique IDs (defined precisely, next) for the location channel. The labeling functions are not known to the robots. The only requirement is that whatever procedure makes the ID assignment does not violate the following property:

Property 1. Labeling functions have:

$$\forall x, y, z \in V, e_y = (x, y) \in E_{ch} \land e_z = (x, z) \in E_{ch} \Rightarrow f_{ch}(e_y) \neq f_{ch}(e_z).$$

Property 1 ensures that each neighborhood will have non-repeating IDs, which we call *locally unique IDs*. In the motivating system, it applies to the location IDs. It also holds for global IDs which are not locally in conflict *i.e.*, no individual robot will see the same wireless ID broadcasted by two different robots in its wireless neighborhood. Note that any labeling that provides unique IDs satisfies property 1.

To examine an individual robot's perspective we introduce several additional concepts.

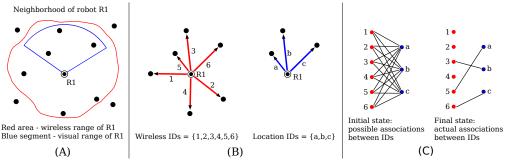


Figure 1: This figure presents a view of a local neighborhood around a robot in a multi-robot system (A) and some of the graphs which are associated with the channel association problem (B),(C). The wireless communication channel (in red) connects robot R1 to some of the robots surrounding it. The camera communication channel (in blue) has a more restricted view of robots in the system. Robot R1 is only aware of the set of IDs which correspond to the connection edges of each channel. The robot represents its knowledge of the association between IDs as a bipartite graph as shown in (C).

Definition 2. A neighborhood is a subset of the system centered on an individual robot i. A neighborhood can be described by the tuple $\langle V^i, E_1^i, E_2^i, f_1, f_2, C_1^i, C_2^i \rangle$ where

- 1. i indicates the robot centered in the neighborhood,
- 2. V^i consists of robots for which a directed edge $e \equiv (v_i, v_j) \in E_1$ exists,
- 3. E_1^i and E_2^i contain all directed edges in E_1 and E_2 which have robot i as source.
- 4. C_1^i , C_2^i are the labels given to edges in E_1^i and E_2^i by f_1 and f_2 , respectively.

Definition 3. The **local view** of robot i is the information directly available to the robot. It can be described by the tuple $\langle n, C_1^i, C_2^i, O^i = \langle O_1^i, O_2^i, ... O_t^i \rangle \rangle$ where

- 1. n is the maximum possible robots in the entire system,
- 2. C_1^i and C_2^i are label sets as previously defined,
- 3. $O^i = \langle O_1^i, O_2^i, ... O_t^i \rangle$ is a sequence of observations (Def. 5) of the robots in the neighborhood of i, and t is the time step during which an observation was made.

Definition 4. An activation captures the ground truth of the communication state of a neighborhood for a given time step t, described by the tuple $A_t^i = \langle C_{1At}^i, C_{2At}^i \rangle$ where

- 1. C_{1At}^i is the set of labels of robots which are in range of and from which a message was sent to robot i on channel 1 during time step t,
- 2. C_{2At}^i is defined similarly for channel 2.

Definition 5. An observation is an individual robot's perception of an activation for a given time step t and is described by the tuple $O_t^i = \langle C_{1Ot}^i, C_{2Ot}^i \rangle$ where

- 1. C_{1Ot}^i is the set of labels of robots from which a message was received by robot i on channel 1 during the time step,
- 2. C_{2Ot}^i is defined similarly for channel 2.

Given the assumptions about no failure and no interruption in messages, activations and their corresponding observations are identical.

Definition 6. Each robot maintains an **information state**, or **i-state**, which represents the current knowledge the robot

has about its neighborhood. The i-state is described by the bipartite graph $S^i = \langle C_1^i, C_2^i, E_{1-2} \rangle$ where

- 1. C_1^i and C_2^i are label sets as previously defined and represent the two distinct vertex sets of the graph,
- 2. E_{1-2} is the edge set of the graph and represents possible associations between elements from each label set.

Each robot starts in complete ignorance of the actual oneto-one association of labels. As observations arrive, it updates its i-state by removing edges from the graph.

Definition 7. The **one-to-one association problem** for each robot *i* is:

<u>Input</u>: a local view with an observation sequence,

Output: a complete or partial one-to-one association between channel 1 labels and channel 2 labels for the label tuple $\langle C_1^i, C_2^i \rangle$ corresponding to i's neighborhood.

Algorithms to Solve the Association Problem

In this section we present three algorithms that can be used to achieve the one-to-one association problem in a multirobot system. The algorithms operate for a system with the following properties:

- 1. There are no communication failures, lost messages or message degradation.
- 2. The channel inclusion property holds.
- 3. The minimum message size for the wireless channel is $\log_2{(n)}$ bits.
- 4. The message size for the visual channel is 1 bit.
- Every robot can receive and process messages from any number of robots in each channel during a given time step.
 Robots can only send one message in each channel during a given time step.

A **listener** is a robot from whose perspective we are analyzing the system. A **message pair** consists of the transmission of a message in each channel by a given robot during the same time step. The earlier assumption that each listener begins knowing the number of robots it can receive messages from in each channel can be achieved by having a single time step in which every robot transmits a message pair.

Naïve Algorithm.

In the Naïve algorithm, every robot emits a visual message during the time step which is equivalent to the robot's wireless ID. A listener simply fills a table associating a wireless ID to a location ID which is currently active. The pseudo code for this algorithm can be seen below.

Algorithm 1 Naïve Algorithm

```
procedure Naïve

matchTable 
ho [wireless ID, location ID] pairs

for t \leftarrow 1, n+1 do 
ho For each possible wireless ID

if t-1 == myID then

broadcast(visualMessage)

end if

if messagesReceived() then

if local\_robot.isActive() then

pair \leftarrow [t-1, locationID]

matchTable.add(pair)

end if

end for
end procedure
```

Logarithmic Algorithm

In the Logarithmic algorithm the fact that n identifiers can be represented with $\log_2{(n)}$ bits is used. The listener associates each observable robot with an array of size $\log_2{(n)}$. At time step t, all robots whose t^{th} bit is equal to 1 will emit a visual message. After each time step a listener can fill one more bit in the array of bits corresponding to the wireless IDs of the visually active robots. The pseudo code is below.

Algorithm 2 Logarithmic Algorithm

```
1: procedure LOGARITHMIC
        self\_bit\_arr \leftarrow to\_bit\_array(my\_ID)
 3:
        table\_arr \leftarrow array[num\_neighbors][log_2(n)]
 4:
        for t \leftarrow 1, \log_2(n) + 1 do
             if self_bit_arr[t-1] == 1 then
 5:
                 broadcast(visualMessage)
 6:
 7:
             end if
 8:
            for i \leftarrow 0, num_neighbors do
 9:
                 if neighbor[j].isActive() then
10:
                     table\_arr[t][j] = 1
11:
                 end if
12:
             end for
13:
         end for
14: end procedure
```

Probabilistic Algorithm

In the Probabilistic algorithm all robots construct an $h \times s$ i-state matrix to represent their i-state. Here h represents the number of robots in direct communication over channel 1 (C_1^i from S^i) and s – the number of robots over channel 2 (C_2^i from S^i). The i-state starts filled with 1's, which indicates complete ignorance of the one-to-one associations. Each cell in the i-state matrix corresponds to an edge in E_{1-2} from S^i . The wireless IDs (labels) of all robots that are within communication range are associated to an activity array (C_{1Ot}^i from O_t^i) that keeps track of which robot

transmitted a message in the current time step (1 is used to denote a transmission). Each wireless ID is given an index $w \in [0,h-1]$. Similarly, the location IDs are associated with an activity array $(C_{2Ot}^i \text{ from } O_t^i)$. Each location ID is given an index $l \in [0,s-1]$. At any moment the i-state matrix contains the listener's current information about the matching between wireless IDs and location IDs.

During each time step, each robot emits a message pair with 50% probability. Every listener fills in an $h \times s$ observation matrix, which captures an observation of the neighborhood made during the current time step. This matrix is filled based on the following principle: The [w][l] entry in the observation matrix is filled with a 1 if both the w^{th} wireless ID and l^{th} location ID were active during the current time step. All other entries are filled with 0's.

Once the observation matrix has been constructed it is used to update the i-state matrix of the listener. All [w][l] entries from the observation matrix for which either the w^{th} wireless ID or l^{th} location ID have been active in the current time step are multiplied with the corresponding i-state matrix entries. This produces a new i-state matrix.

Once the i-state matrix contains only s 1's, the one-to-one association problem for the listener and its neighborhood is complete. The non-zero [w][l] entries tell us that the w^{th} wireless ID and the l^{th} location ID belong to the same robot.

Pseudo code appears in **Algorithm 3**, and an i-state and observations for a single run of the algorithm for one neighborhood are depicted in Fig. 2.

Algorithm 3 Probabilistic Algorithm

```
1: procedure Probabilistic
 2:
        broadcast(messagePair)
 3:
                                  > number of wireless IDs in range
 4:
                                  ⊳ number of location IDs in range
 5:
        istate[h][s] \leftarrow 1's
 6:
        observ[h][s]
 7:
        wifi\_activity[h]
                                             ⊳ 0 - inactive, 1 - active
 8:
        vis\_activity[s]
                                             \triangleright 0 - inactive, 1 - active
 9:
        for each time step do
10:
            if fairCoinFlip() then
                broadcast(messagePair)
11:
12:
            wifi\_activity \leftarrow receivedWifiMsqs()
13:
14:
            vis\_activity \leftarrow receivedVisMsgs()
            observ \leftarrow fillObs(wifi\_activity, vis\_activity)
15:
16:
            istate \leftarrow updateState(observ)
            if istate.isFinal() then
17:
18:
                ready \leftarrow TRUE
19:
            end if
20:
        end for
21: end procedure
```

Information Space Interpretation

Although only the probabilistic algorithm explicitly uses the idea of an information space, both deterministic algorithms have an information space interpretation. In this section we examine the information space explored by the algorithms.

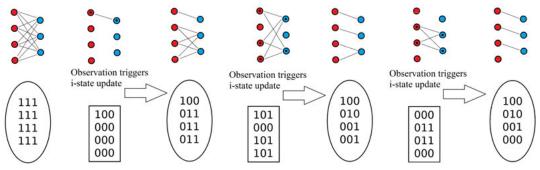


Figure 2: An i-state path through the transition graph. Here we can see the i-state of the robot (the matrices in ovals) and its progression as the robot makes observations (the matrices in squares). Bipartite graphs corresponding to each i-state and observation appear at the top.

Structure of Individual I-States and Observations

As observations about the neighborhood are made, a listener can update its i-state accordingly. The initial i-state is an $h \times s$ complete bipartite graph. An observation partitions the i-state bipartite graph into active and inactive vertices. All edges between an active and inactive vertex are eliminated from the i-state graph. The partitioned vertices form bipartite sub-graphs. Once such a partitioning occurs, the listener has reduced the potential associations between wireless IDs and location IDs. Every subsequent observation can split the already existing partitions further.

An observation can partition the vertices in such a way that it brings no additional information about the neighborhood. There are two observations which always bring no information: all robots are active or no robots are active. These are the zero-information observations. Observations can be paired based on symmetry – the partitioning these pairs induce on the i-state bipartite graph is the same.

The final i-state is a disconnected bipartite graph which is composed of bipartite sub-graphs of size 2 or 1. The sub-graphs of size 2 indicate a one-to-one association between a wireless ID and a location ID while the sub-graphs of size 1 show that a wireless ID belongs to a robot which cannot be observed by the listener.

The number of i-states for a given neighborhood is finite and can be found by calculating Bell's number for a set of size h. Bell's number counts the number of ways a set of a given size can be partitioned in.

The number of observations for a given neighborhood is 2^h . Since we have assumed an activation probability for a robot to be 50% (during the probabilistic algorithm), all neighborhood activations and corresponding observations have the same probability of occurring.

I-State Transition Graph

The i-states and observations for a given neighborhood form the **directed i-state transition graph**. The i-states form the vertex set of the transition graph and the directed edges correspond to the set of observations which lead from one i-state to another. A version of the transition graph can be seen in Fig. 3. The figure shows all i-states for a 4×3 neighbor-

hood (4 wireless IDs and 3 location IDs), all i-state to i-state transitions, and the number of observation for each transition.

The graph highlights properties of the i-state space:

- 1. The i-states can be grouped into several categories based on the number of partitions of the initial i-state bipartite graph they contain. We call these categories **levels**. There are h levels in each transition graph corresponding to same size partitions of the graph S^i . The initial i-state is at level 1 (trivial partition of the graph S^i), and the final i-state is at level h (there are h partitions of the graph S^i). Any intermediate level will have between 2 and h-1 partitions of S^i .
- At each next level the number of zero-information states doubles.
- 3. At each level the transition probability is equal for all transitions (for the probabilistic algorithm).
- 4. Each i-state can transition to itself (through zero-information observations).
- 5. There are no transitions between i-states from a higher partition level to a lower partition level and inside a level (with the exception of self-transitions).

Deterministic Algorithms and the I-State Graph

The Na $\ddot{\text{u}}$ ve algorithm uses only observations resulting from a single active robot. This restricts the paths through the transition graph to a subset of the 1-level transitions. The Logarithmic algorithm, on the other hand, allows any n-step path through the transition graph. The specific path for both algorithms is predetermined once a neighborhood is formed.

Algorithm Performance Analysis

After examining the information space we can examine the performance of the three algorithms presented earlier.

Naïve Algorithm

The Naïve algorithm runs for n time steps, where n is the number of unique IDs that can be used by the system of robots. The algorithm runs for n time steps regardless of how many of the n identifiers are in use by actual robots. Every agent emits only one message during the execution of this algorithm – at most n for the entire system.

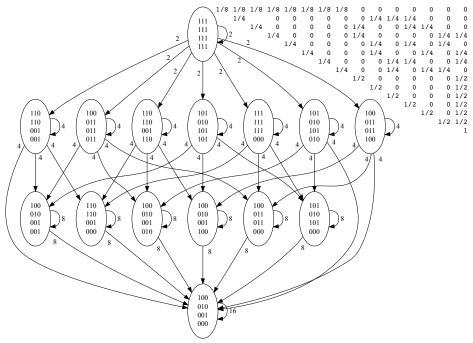


Figure 3: I-state transition graph and corresponding transitional probability matrix for the case of h=4 wireless, s=3 location IDs. Each graph node contains an $h \times s$ state matrix and each edge is labeled with the number of observations which trigger the given transition.

Logarithmic Algorithm

The Logarithmic algorithm runs for $\log_2{(n)}$ time steps and generates at most $\frac{1}{2}n\log_2{(n)}$ messages. This is due to the fact that at most half of all robots will emit a message pair at each time step, since only half of all possible unique IDs will have a 1 in their i^{th} bit.

Probabilistic Algorithm

The one-to-one association problem for a neighborhood of size $h \times s$ has a lower bound of $ceil(\log_2{(s)})$ time steps. The best case scenario for the association problem occurs when each observation made by the listener splits all current partitions of the i-state bi-graph S^i evenly.

Due to the probabilistic nature of this algorithm, its overall analysis is not straightforward and a closed-form solution for its expected running time has not been discovered.

Mathews et al. (2010) present a probabilistic algorithm which can be modeled as a branching process and show an approximate upper bound for the mean of the expected time. Robots participating in the algorithm solving this model "die-off". The authors emphasize the difficulty in finding a closed form solution for the probability density function of the process.

We were unable to apply this model to the more general system we present here. The difficulty arises from successfully identifying elements of our model which can be regarded as "individuals" in a population which can be discarded (produce zero offspring).

Another approach is to use the **transition probability matrix** corresponding to the directed i-state transition graph.

The graph has the necessary properties of an absorbing Markov chain (Grinstead and Snell, 1997) — it has an absorbing state (the final state) and this state can be reached from any other state in the graph (transient states). See Fig. 3 for an example of the transition probability matrix. The expected number of steps from any state to reaching the final state can be calculated using the following formula:

$$\mathbf{t} = (I - Q)^{-1} \mathbf{1} \tag{1}$$

Here Q is a $t \times t$ matrix containing all transient to transient transition probabilities, \mathbf{t} denotes a vector of size t, I is the identity matrix, and $\mathbf{1}$ is a column vector of all 1's. The i^{th} element of \mathbf{t} gives the average expected number of steps to absorption if the process started from the i^{th} transient state. Thus, Eq. (1) can be used to calculate the expected running time to complete the one-to-one association problem for a given neighborhood.

This approach, however, is prohibitively expensive for all but the smallest values of h. As was noted earlier, the number of i-states (t) for a given neighborhood is given by Bell's number: B_h . Berend and Tassa (2010) give an upper bound of Bell's number which is O(h!) and $O(2^h)$. A simple application of all possible observations to all possible i-states can generate the matrix in $O(2^hB_h)$ time. The matrix inversion needed in Eq. (1) will take $O(B_h^3)$. The absorbing Markov chain method, therefore, runs in $O(h!^3)$.

A third approach is to simulate the running of the algorithm and gather statistical data. This was done by simulating activations of a neighborhood and running the Probabilistic algorithm until a one-to-one association was made

or a threshold was reached. Each neighborhood was run between 50,000 and 100,000 times. We simulated all neighborhood size combinations of $h \in [2,85]$ and $s \in [1,h-1]$. To validate the date from the simulation we also calculated the precise expected value for all combinations of $h \in [2,8]$ and $s \in [1,h-1]$ using the absorbing Markov chain approach. The absorbing Markov chain method results matched the simulation results up to the precision we used during the calculations (0.0001 time steps).

The results from the simulation trials of neighborhoods up to size 40 can be seen in Fig. 4, the remaining trial data is omitted to keep the figure more readable.

The results from the simulation can be informally approximated (further work is needed to get a good fit for the data) by the following two functions which give the expected time steps to complete the one-to-one association problem:

- 1. $2\log_2(h)\log_2(s+1)$
- 2. $\operatorname{arsinh}(hs)$

Using the first function, we can approximate the number of expected messages a given robot will emit: $0.5 \times 2\log_2{(h)}\log_2{(s+1)}$. The h and s would be based on the longest lasting neighborhood the robot participates in and assumes that the robots stop transmitting messages once all their neighbors inform them that they have completed the one-to-one association. For a dynamic system, this assumption would not be true and the robot may continue transmitting messages after the initial association has been completed.

Algorithm Comparison

The two deterministic algorithms have a performance which is dependent on the global size of the system. They also provide an exact execution time for the one-to-one association problem. The Naïve algorithm gives us the slowest performance but uses the least amount of messages. The Logarithmic algorithm is faster than the Naïve one but requires an increase in the number of messages used. The two deterministic algorithms also require very minimal use of the wireless channel (only used during the first system wide transmission, which is not strictly required for the Logarithmic algorithm).

The probabilistic algorithm does not depend on the global size of the system. Every robot can quickly achieve the one-to-one association for its neighborhood. This algorithm has the disadvantage that it has no guaranteed time expectancy. The overall performance of the system is dependent directly on the types of neighborhoods that compose it so an analysis of the robot distribution may be required before choosing this algorithm.

Relaxing Assumptions and Extending the Formalism

In this section we discuss relaxing some of the assumptions introduced in the course of the paper. We show that after

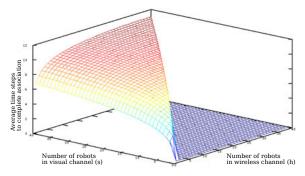


Figure 4: This is a plot of the average number of time steps to complete the association problem for all $h \times s$ neighborhoods for which $h \in [2,40]$ and $s \in [1,h-1]$. Every neighborhood was simulated between 50,000 and 100,000 times to obtain the results

relaxing these assumptions the formalism still applies and the algorithms can still be used with minor modifications.

We have assumed that robots operate in an environment with no noise, communication errors or failures. This assumption guarantees that when a robot sends a message it is always received. This results in edges in the i-state (possible associations) being either 0 or 1. If we allowed noise, communication errors and failures, an observation may carry false positives or false negatives. The formalism and probabilistic algorithm can still handle this case. However, instead of a 0 or 1 stored as a possible association between two IDs, an appropriate statistical measure is computed and stored for each possible association. This makes the bipartite graph representing an i-state a weighted one. The finite information space interpretation presented in this paper is not directly applicable to this case.

The channel inclusion property imposes restrictions on how the coverages of the communication channels are related. The assumption in this paper guarantees that a robot's continued observation will result in an i-state in which all robots in visual (channel 2) range will be matched to robots in wireless (channel 1) range. This restricts the possible final i-states. If the assumption is relaxed fully, the resulting final i-state may be an empty i-state (a fully disconnected bipartite graph) indicating that the sets of robots in each channel are completely disjoint. The formalism will still correctly represent the system. Perhaps more surprisingly, all three algorithms correctly handle this situation with no modification other than a modified stopping condition.

We specify that there are two communication channels. The addition of more channels can be beneficial for the Probabilistic algorithm, as it can partition the i-state bipartite graph faster. This, however, would require the i-state's dimensions to grow as the number of channels. In this work we have not pursued any analysis on the impact on performance that the addition of channels will have.

The visual channel is assumed to be a binary channel. If the visual channel can distinguish between several colors, as done in (Mathews et al., 2010), the effective bit-rate of the visual channel increases as the color can be used to pass additional information. For example, the message pair can contain a colored bit in the visual channel and a color label in the broadcast channel. This will result in a faster partitioning of the i-state bipartite graph.

Property 1 allows robots communicating through a communication channel to share IDs. For example, a system in which robots are guaranteed to not move in a way that will bring two duplicate IDs in the same neighborhood can operate with no conflicts. A system where positions can change arbitrarily will require the stricter unique ID requirement. If an outside reference, such as geographical data or some environmental property, is used to assign IDs a system can have some reuse of IDs and a guarantee that a conflict will not occur.

Conclusion

In this paper we consider an association problem between IDs in two independent communication channels in a multirobot system. The first channel is a radio channel which carries indexical knowledge about the robots while the second channel is a situated one and carries objective knowledge about the robots. Robots in the system need to make a one-to-one association between the two types of IDs as a preprocessing step to solving system tasks.

We provide a detailed formal description of the system and examine its properties from an information filtering perspective. We also show that this problem can be solved quickly and without the need for global localization knowledge and global communication by describing and analyzing two deterministic and one probabilistic algorithms.

The analysis of the probabilistic algorithm showed us that a closed-form solution for the expected performance of the algorithm based on the absorbing Markov chain is prohibitively expensive and that the applicability of another — based on branching processes — is uncertain but needs to be considered in more depth. We provide simulation results and give two functions which provide an informal approximation to the algorithm's performance. We compare the three algorithms and show that each has its advantages and drawbacks, relating to speed of execution, messages sent and communication channels used.

Lastly, we examined the ability of the formalism and algorithms we described to handle more realistic scenarios by relaxing some simplifying assumptions made during the creation of the formalism and analysis of the algorithms.

One future extension of this work is to examine in more detail the applicability of branching processes to the system model defined in this paper. Another extension is to analyze the performance of the algorithms after some assumptions, such as no communication failures, have been relaxed. A third extension is finding a better fit to the simulation data which can be used to predict the behavior of larger multirobot systems operating under the probabilistic algorithm.

References

- Agre, P. E. and Chapman, D. (1987). Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence Volume 1*, AAAI'87, pages 268–272. AAAI Press.
- Berend, D. and Tassa, T. (2010). Improved bounds on Bell numbers and on moments of sums of random variables. *Probability and Mathematical Statistics*, 30(2):185–205.
- Cassinis, R. and Tampalini, F. (2007). AMIRoLoS an active marker internet-based robot localization system. *Robotics and Autonomous Systems*, 55(4):306 315.
- Dieudonné, Y., Dolev, S., Petit, F., and Segal, M. (2009). Deaf, dumb, and chatting asynchronous robots. In *Proceedings* of the 13th International Conference on Principles of Distributed Systems, OPODIS '09, pages 71–85, Berlin, Heidelberg. Springer-Verlag.
- Fox, D., Burgard, W., Kruppa, H., and Thrun, S. (2000). A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, 8(3):325–344.
- Franchi, A., Oriolo, G., and Stegagno, P. (2009). Mutual localization in a multi-robot system with anonymous relative position measures. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3974–3980.
- Garrido-Jurado, S., Muñoz Salinas, R., Madrid-Cuevas, F. J., and Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recogn.*, 47(6):2280–2292.
- Grinstead, C. M. and Snell, J. L. (1997). Introduction to Probability, chapter 11: Markov Chains. American Mathematical Society.
- Haccou, P., Jagers, P., and Vatutin, V. (2007). Branching Processes: Variation, Growth, and Extinction of Populations. Cambridge Studies in Adaptive Dynamics.
- Howard, A., Parker, L. E., and Sukhatme, G. S. (2004). The sdr experience: Experiments with a large-scale heterogenous mobile robot team. In 9th International Symposium on Experimental Robotics 2004, Singapore.
- LaValle, S. M. (2009). Tutorial: Filtering and planning in information space. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS'09, pages 1–1, Piscataway, NJ, USA. IEEE Press.
- Lespérance, Y. and Levesque, H. J. (1994). Indexical knowledge and robot action - a logical account. Artificial Intelligence, 73:69–115.
- Mathews, N., Christensen, A. L., Ferrante, E., O'Grady, R., and Dorigo, M. (2010). Establishing spatially targeted communication in a heterogeneous robot swarm. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 Volume 1*, AAMAS '10, pages 939–946, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.
- Støy, K. (2001). Using situated communication in distributed autonomous mobile robotics. In In Proceedings of the 7th Scandinavian conf. on Artificial Intelligence.

Comparison of Selection Methods in On-line Distributed Evolutionary Robotics

Iñaki Fernández Pérez, Amine Boumaza and François Charpillet

Université de Lorraine, INRIA, Campus scientifique BP 239 Vandoeuvre-lès-Nancy Cedex, F-54506, firstname.lastname@loria.fr

Abstract

In this paper, we study the impact of selection methods in the context of on-line on-board distributed evolutionary algorithms. We propose a variant of the mEDEA algorithm in which we add a selection operator, and we apply it in a task-driven scenario. We evaluate four selection methods that induce different intensity of selection pressure in a multi-robot navigation with obstacle avoidance task and a collective foraging task. Experiments show that a small intensity of selection pressure is sufficient to rapidly obtain good performances on the tasks at hand. We introduce different measures to compare the selection methods, and show that the higher the selection pressure, the better the performances obtained, especially for the more challenging food foraging task.

Introduction

Evolutionary robotics (ER) (Nolfi and Floreano (2000)) aims to design robotic agents' behaviors using evolutionary algorithms (EA) (Eiben and Smith (2003)). In this context, EA's are traditionally seen as a tool to optimize agents' controllers w.r.t to an explicit objective function (fitness). This process is carried out in an off-line fashion; once the behavior is learned and the controller optimized, the agents are deployed and their controllers' parameters remain fixed.

On the other hand, on-line evolution (Watson et al. (2002)) takes a different approach in which behavior learning is performed during the actual execution of a task. In these algorithms, learning or optimization is a continuous process, *i.e.* robotic agents are constantly exploring new behaviors and adapting their controllers to new conditions in their environment. Usually, this is referred to as adaptation.

These two visions of ER can be related to on-line and offline approaches in Machine Learning (ML). Off-line ML algorithms learn a specific task and solutions should generalize to unseen situations after the learning process, whereas on-line ML algorithms progressively adapt solutions to new presented situations while solving the task. In this sense, both on-line ML algorithms and on-line EA's perform lifelong adaptation or learning, to possibly changing environments or objectives. In this paper, we focus on on-line distributed evolution of swarm robotic agents. We are interested in learning individual agents behaviors in a distributed context where the agents adapt their controllers to environmental conditions independently while deployed. These agents may locally communicate with each other and do not have a global view of the swarm. In this sense, this approach finds many ties with Artificial Life, where the objective is to design autonomous organisms that adapt to their environment.

On-line distributed evolution may be viewed as distributing an EA on the swarm of agents. Traditional evolutionary operators (mutation, crossover etc.) are performed on the agents and local communication ensures the spread of genetic material in the population of agents.

In EA's, selection operators drive evolution toward fit individuals by controlling the intensity of selection pressure to solve the given task. These operators and their impact on evolutionary dynamics have been extensively studied in off-line contexts (Eiben and Smith (2003)). In this paper, we study their impact in on-line distributed ER, where evolutionary dynamics are different to the off-line case: selection is performed locally on partial populations and fitness values on which selection is performed are not reliable. Our experiments show that, in this context, a strong selection pressure leads to the best performances, contrary to classical approaches in which lower selection pressure is preferred, to maintain diversity in the population. This result suggests that, in distributed ER algorithms, diversity is already maintained by the disjoint sub-populations.

Several authors have addressed on-line evolution of robotic agent controllers in different contexts: adaptation to dynamically changing environments (Dinu et al. (2013)), parameter tuning (Eiben et al. (2010)), evolution of self-assembly (Bianco and Nolfi (2004)), communication (Pineda et al. (2012)), phototaxis and navigation (Karafotias et al. (2011), Silva et al. (2012)). Some of this work is detailed in the next section. The authors use different selection mechanisms inducing different intensities of selection pressure to drive evolution. In this paper, we compare different selection operators and measure the impact they have on the

performances of learning two swarm robotics tasks: navigation with obstacle avoidance and collective food foraging.

We begin by reviewing different selection schemes proposed in the context of on-line distributed ER and then we present the algorithm that will serve as a test bed, along with the selection methods we compare. In the fourth section, we detail our experimental setting and discuss the results. Finally, we close with some concluding remarks and future directions of research.

Related Work

In the following, we review several on-line distributed ER algorithms and discuss the selection mechanisms that were applied to ensure the desired intensity of selection pressure in order to drive evolution.

A common characteristic of on-line distributed ER algorithms is that each agent has one controller at a time, that it executes (the active controller), and locally spreads altered copies of this controller to other agents. In this sense, agents have only a partial view of the population in the swarm (a local repository). Fitness assignment or evaluation of individual chromosomes is performed by the agents themselves and is thus noisy, as different agents evaluate their active controllers in different conditions. Selection takes place when the active controller is to be replaced by a new one from the repository.

PGTA (Probabilistic Gene Transfer Algorithm) introduced by Watson et al. (2002), is commonly cited as the first implementation of a distributed on-line ER algorithm. This algorithm evolves the weights of fixed-topology neural controllers and agents exchange parts (genes) of their respective chromosomes using local broadcasts. The algorithm considers a virtual energy level that reflects the performance of the agent's controller. This energy level increases every time the agents reach an energy source and decreases whenever communication takes place. Furthermore, the rate at which the agents broadcast their genes is proportional to their energy level and conversely, the rate at which they accept a received gene is inversely proportional to their energy level. This way, selection pressure is introduced in that fit agents transmit their genes to unfit ones.

Silva et al. (2012) introduced odNEAT, an on-line distributed version of NEAT (Neuro-Evolution of Augmenting Topologies) (Stanley and Miikkulainen (2002)), where each agent has one active chromosome that is transmitted to nearby agents. Collected chromosomes from other agents are stored in a local repository within niches of species according to their topological similarities, as in NEAT. Each agent has a virtual energy level that increases when the task is performed correctly and decreases otherwise. This energy level is sampled periodically to measure fitness values and, whenever this level reaches zero, the active chromosome is replaced by one in the repository. At this point, a species is selected based on its average fitness value, then a chro-

mosome is selected within this species using binary tournament. Each agent broadcasts its active chromosome at a rate proportional to the average fitness of the species it belongs to. This, added to the fact that the active chromosome is selected from fit niches, maintains a certain selection pressure toward fit individuals.

EDEA (Embodied Distributed Evolutionary Algorithm) (Karafotias et al. (2011)), was applied to different swarm robotics tasks: phototaxis, navigation with obstacle avoidance and collective patrolling. In this algorithm, each agent possesses one chromosome, whose controller is executed and evaluated on a given task. At each iteration, agents broadcast their chromosomes alongside with their fitness to other nearby agents with a given probability (fixed parameter). Upon reception, an agent selects a chromosome from those collected using binary tournament. This last chromosome is then mutated and recombined (using crossover) with the current active chromosome with probability $\frac{f(x')}{s_c \times f(x)}$, where f(x') is the fitness of the selected chromosome, f(x)is the fitness of the agent's current chromosome and s_c is a scalar controlling the intensity of selection pressure. To ensure an accurate measure of fitness values, agents evaluate their controllers for at least a minimum period of time (maturation age), during which agents neither transmit nor receive other chromosomes.

With mEDEA (minimal Environment-driven Distributed Evolutionary Algorithm), Bredeche and Montanier (2010) address evolutionary adaptation with implicit fitness, *i.e.* without a task-driven fitness function. The algorithm takes a gene perspective in which successful chromosomes are those that spread over the population of agents and which requires: 1) to maximize mating opportunities and 2) to minimize the risk for agents (their vehicles).

At every time step, agents execute their respective active controllers and locally broadcast mutated copies of the corresponding chromosomes. Received chromosomes (transmitted by other agents) are stored in a local list. At the end of the execution period (*lifetime*), the active chromosome is replaced with a randomly selected one from the agent's list and the list is emptied. An agent dies if there are no chromosomes in its list (if it did not meet other agents) and it remains dead until it receives a chromosome from another agent passing by.

The authors show that the number of living agents rises with time and remains at a sustained level. Furthermore, agents develop navigation and obstacle avoidance capabilities that allow them to better spread their chromosomes. This work shows that environment-driven selection pressure alone can maintain a certain level of adaptation in a swarm of robotic agents. A slightly modified version of this algorithm is used in this work and is detailed in the next section.

Noskov et al. (2013) proposed MONEE (Multi-Objective aNd open-Ended Evolution), an extension to mEDEA adding a task-driven pressure as well as a mechanism (called

market) for balancing the distribution of tasks among the population of agents, if several tasks are to be tackled. Their experiments show that MONEE is capable of improving mEDEA's performances in a collective concurrent foraging task, in which agents have to collect items of several kinds.

The authors show that the swarm is able to adapt to the environment (as mEDEA ensures), while foraging different kinds of items (optimizing the task-solving behavior). In this context, each type of item is considered a different task. The algorithm uses an explicit fitness function in order to guide the search toward better performing solutions. The market mechanism, which takes into account the scarcity of items, ensures that agents do not focus on the most frequent kind of items (the easiest task), thus neglecting less frequent ones. In their paper, the agent's controller is selected using rank-based selection from the agent's list of chromosomes. The authors argue that when a specific task is to be addressed, a task-driven selection pressure is necessary. This idea is discussed in the remainder of this paper.

In the aforementioned works, authors used different classical selection operators from evolutionary computation in on-line distributed ER algorithms. It is however not clear if these operators perform in the same fashion as when they are used in an off-line non-distributed manner. In an on-line and distributed context, evolutionary dynamics are different, since selection is performed locally at the agent level and over the individuals whose vehicles had the opportunity to meet. In addition, and this is not inherent to on-line distributed evolution but to many ER contexts, fitness evaluation is intrinsically noisy as the agents evaluate their controllers in different conditions, which may have a great impact on their performance. A legitimate question one could ask is: does it still make sense to use selection?

In this paper, we compare different selection methods corresponding to different intensities of selection pressure in a task-driven context. We apply these methods in a modified version of mEDEA and measure their impact on two different swarm robotics tasks.

Algorithms

In this section, we describe the variant of mEDEA we used in our experiments (Algorithm 1). It is run by all the agents of the swarm independently in a distributed manner. At any time, each agent possesses a single controller which is randomly initialized at the beginning of evolution.

The main difference w.r.t. mEDEA is that the algorithm alternates between two phases, namely an evaluation phase, in which the agent runs, evaluates and transmits its controller to nearby listening agents, and a listening phase, in which the agent does not move and listens to incoming chromosomes, sent by nearby agents. The evaluation and the listening phases last $T_{\rm e}$ and $T_{\rm l}$ respectively, and, for different robots, they take place at different moments. Since the different robots are desynchronized, robots in the evaluation

phase are able to spread their genomes to other robots that are in the listening phase.

If only one common phase takes place, an agent that turns on the spot transmits its controller to any fitter agent crossing it, as broadcast and reception are simultaneous. This separation in two phases is inspired from MONEE where it is argued that it lessens the spread of poorly achieving controllers. Also, task-driven selection was introduced in MONEE to simultaneously tackle several tasks.

The agent's controller is executed and evaluated during the evaluation phase. For each agent, this phase lasts $T_{\rm e}$ time-steps at most 1. During this phase, at each time-step the agent executes its current controller by reading the sensors' inputs and computing the motors' outputs. The agent also updates the fitness value of the controller, based on the outcome of the its actions, and locally broadcasts both the chromosome corresponding to its controller and its current fitness value.

Once the $T_{\rm e}$ evaluation steps are elapsed, the agent begins its listening phase, which lasts T_1 time-steps. During this phase, the agent stops and listens for incoming chromosomes from nearby passing agents (agents that are in their evaluation phase). These chromosomes are transmitted along with their respective fitness values. Consequently, at the end of this phase, an agent has a local list of chromosomes and fitnesses, or local population. Another difference w.r.t mEDEA is that the local population also contains the agent's current chromosome. This is done to ensure that all agents always have at least one chromosome in their respective populations, which happens particularly when an agent is isolated during its listening phase and does not receive any other chromosome. In mEDEA, isolated agents stay inactive until they receive a chromosome from another agent passing by.

After the listening period, the agent needs to load a new controller for its next evaluation phase. To do so, it selects a chromosome from its list using one of the selection methods discussed further. The selected chromosome is then mutated and becomes the agent's active controller. In this case, mutation consists in adding a normal random variable with mean 0 and variance σ^2 to each gene (each synaptic weight of the neuro-controller).

Once the next controller is chosen, the list is emptied. This means selection is performed on a list of chromosomes that have been collected by the agent during the previous listening phase. At this time, the new controller's evaluation phase begins. We consider one iteration of the algorithm (evaluation plus listening phase) as one generation.

The selection method selects the new chromosome among the collected ones based on their fitness. This can be done in different manners, depending on the desired intensity of selection pressure. In this paper we compare four different

 $^{^{1}}$ A little random number is substracted from $T_{\rm e}$ so as the evaluation phases of the agents are not synchronized.

Algorithm 1 mEDEA

```
1: g_a := random()
 2: while true do
      1 := \emptyset
 4:
      // Evaluation phase
 5:
       for t=1 to T_{\rm e} do
 6:
         exec(g_a)
 7:
         broadcast(g_a)
 8:
       end for
 9:
       // Listening phase
10:
       for t = 1 to T_1 do
11:
         l := l \cup listen()
12:
       end for
13:
       1 := 1 \bigcup \{g_a\}
14:
       selected := select(1)
15:
       g_a := mutate(selected)
16: end while
```

selection methods, each one defining a different intensity of task-driven selection pressure. The choice of these selection methods aims at giving a large span of intensities of selection pressure, from the strongest (*Best*), to the lowest (*Random*):

Best Selection: This method deterministically selects the controller with the highest fitness. This is the selection method with the strongest selection pressure, as the agent will never be allowed to select a controller with a lower fitness than the previous one. *Best* selection can be compared to an elitist selection scheme where the best fit controllers are always kept.

Algorithm 2 Best selection

```
1: order x_i and index as x_{i:n} such that: f(x_{1:n}) \ge f(x_{2:n}) \ge \ldots \ge f(x_{n:n})
2: return x_{1:n}
```

Rank-Based Selection: In this case, selection probabilities are assigned to each controller according to their rank, *i.e.* the position of the controller in the list, once sorted w.r.t. fitness values. The best controller has the highest probability of being selected; however, less fit controllers still have a positive probability of being selected. Traditionally, this method is preferred to Roulette Wheel selection that assigns individuals probabilities proportional to their fitness values, which highly biases evolution toward best individuals.

Binary Tournament: This method uniformly samples a number of controllers equal to the size of the tournament (two in our case) and selects the one with the highest fitness. Here, the selection pressure is adjusted through the size of

Algorithm 3 Rank-based selection

```
1: order x_i and index as x_{i:n} such that: f(x_{1:n}) \geq f(x_{2:n}) \geq \ldots \geq f(x_{n:n})

2: select x_{i:n} with probability Pr(x_{i:n}) = \frac{n+1-i}{1+2+\ldots+n}

3: return x_{i:n}
```

the tournament: the higher the size, the higher the selection pressure, the extreme case being when the tournament size is equal to the size of the population. In this case, the best controller is chosen². Conversely, when the size of the tournament is two, the induced selection pressure is the lowest.

Algorithm 4 k-Tournament selection

```
1: uniformly sample k x_i, noted \{x_{1:k}, \ldots, x_{k:k}\}

2: order x_{i:k} such that: f(x_{1:k}) \geq f(x_{2:k}) \geq \ldots \geq f(x_{k:k})

3: return x_{1:k}
```

Random Selection: This method selects a controller in the local population at random, disregarding its fitness value and therefore inducing no task-driven selection pressure at all. *Randon* selection is considered as a baseline for comparisons with the other methods that effectively induce a certain task-driven selection pressure. As discussed in the previous section, this is the selection operator used by mEDEA for evolving survival capabilities of the swarm without any task-driven explicit goal. By considering *Random* in our experiments, we aim to compare the original mEDEA selection scheme with more selective operators.

Each one of these four selection methods induces a different intensity of selection pressure on the evolution of the swarm. In the next section, we describe our experiments comparing the impact of each one of these intensities.

Experiments

We compare these selection methods on a set of experiments in simulation for two different tasks, fast-forward navigation and collective foraging, which are two well-studied benchmarks in swarm robotics. All our experiments were performed on the RoboRobo simulator (Bredeche et al. (2013)).

Description

In all experiments, a swarm of robotic agents is deployed in a bounded environment containing static obstacles (black lines in Figure 1). Agents also perceive other agents as obstacles.

All the agents in the swarm are morphologically homogeneous, *i.e.* they have the same physical properties, sensors and motors, and only differ in the parameters of their respective controllers. Each agent has 8 obstacle proximity sensors

²It is assumed that sampling is performed without replacement.

evenly distributed around the agent, and 8 food item sensors are added in the case of the foraging task. An item sensor measures the distance to the closest item in the direction of the sensor. These simulated agents are similar to Khepera or e-puck robots.

We use a recurrent neural network as the architecture of the neuro-controllers of the agents (Figure 1). The inputs of the network are the activation values of all sensors and the 2 outputs correspond to the translational and rotational velocities of the agent. The activation function of the output neurons is a hyperbolic tangent, taking values in [-1, +1]. Two bias connections (one for each output neuron), as well as 4 recurrent connections (previous speed and previous rotation for both outputs) are added. This setup yields 22 connection weights for the navigation task and 38 for the foraging task in the neuro-controller. The chromosome of the controller is the vector of these weights. Table 1 summarizes the different parameters used in our experiments.

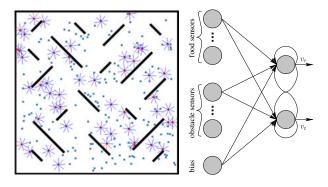


Figure 1: Left: the simulation environment containing agents (red dots with thin hairlines representing sensors), obstacles (dark lines) and food items (blue dots). Right: the architecture of the neuro-controller.

In the navigation task, agents must learn to move as fast and straight as possible in the environment while avoiding obstacles, whereas in the foraging task, agents must collect food items present in the environment (Figure 1). An item is collected when an agent passes over it, at which time it is replaced by another item at a random location.

We define the fitness function for the navigation task after the one introduced in (Nolfi and Floreano (2000)). Each agent r computes its fitness at generation g as:

$$f_r^g = \sum_{t=1}^{T_e} v_t(t) \cdot (1 - |v_r(t)|) \cdot min(a_s(t))$$
 (1)

where $v_t(t)$, $v_r(t)$ and $a_s(t)$ are respectively the translational velocity, the rotational velocity and the activations of the obstacle sensors of the agent at each time-step t of its evaluation phase.

In the foraging task, a controller's fitness is computed as the number of items collected during its evaluation phase.

	1 &									
Experiments										
Number of food items	150									
Swarm size	50 agents									
Exp. length	5×10^5 sim. steps									
Number of runs	30									
Evolution										
Evolution length	~ 250 generations									
T_e	2000 - rand(0, 500) sim. steps									
T_l	200 sim. steps									

Table 1: Experimental settings.

Furthermore, since we are interested in the performance of the entire swarm, we define the swarm fitness as the sum of the individual fitness of all agents at each generation:

$$F_s(g) = \sum_{r \in swarm} f_r^g \tag{2}$$

Nav.: 22, Forag.: 38

 $\sigma = 0.5$

Measures

Chromosome size

Mutation step-size

A characteristic of on-line ER is that agents learn as they are performing the actual task in an open-ended way. In this context, the best fitness ever reached by the swarm is not a reliable measure, since it only reflects a "good" performance at one point of the evolution. Furthermore, fitness evaluation is inherently noisy, due to different evaluation conditions encountered by the agents. Therefore, we introduce four measures that will be used to compare the impact of the different selection methods. These measures summarize information on the swarm spanning over several generations. They are used only for evaluation and comparison of the selection methods and are computed once the evolution has ended. A pictorial description of these four measures is shown in Figure 2.

- Average accumulated swarm fitness (f_c) : is the average swarm fitness in the last generations. This metric reflects the performance of the swarm at the end of the evolution. In our experiments, we compute the average over the last 8% generations.
- Fixed budget swarm fitness (f_b) : is the swarm fitness reached at a certain generation (computational budget). This measure helps to compare different methods on the same grounds. In our experiments, we measure this value at 92% of the evolution, which corresponds to the first generation considered in the computation of f_c .
- Time to reach target (g_f) : is the first generation at which a predefined target fitness is reached. If this level is never reached, g_f corresponds to the last generation. We fixed the target at 80% of the maximum fitness reached over

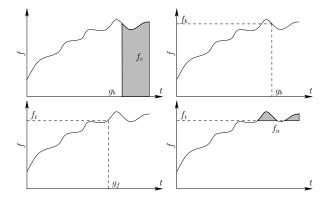


Figure 2: A pictorial description of the four comparison measures. From top to bottom and left to right: the average accumulated swarm fitness, the fixed budget swarm fitness, the time to reach target and the accumulated fitness above target.

all runs and all selection methods. This metric reflects a certain convergence rate of the algorithms, *i.e.* how fast the swarm hits the target fitness on the task at hand.

• Accumulated fitness above target (f_a) : is the sum of all swarm fitness values above a predefined target fitness. It reflects to which extent the target level is exceeded and if this performance is maintained over the long run. We used the same target fitness as with g_f .

These comparison measures are not to be taken individually. For instance f_c and f_b complement each other and give an indication of the level and stability of the performance reached by the swarm at the end of evolution. If f_b and f_c are close then performance of the swarm is stable. Also, g_f and f_a combined reflect how fast a given fitness level is reached and to which extent that level is exceeded. Adding the two latter measures to f_c shows if that trend is maintained.

Results and discussion

For both navigation and foraging tasks, we run 30 independent runs for each selection method, and we measured F_s at each generation in all runs. Figures 3 and 4 show the median F_s per generation over the 30 runs for each task. We computed the four performance metrics in the case of navigation (Figure 5) and of foraging (Figure 6). For both tasks, we performed pairwise³ Mann-Whitney tests at 99% confidence on these measures, between the four selection methods.

On the one hand, upon analysis of Figure 3 and Figure 4, we observe that the swarm rapidly reaches a high fitness level in both tasks whenever there is a task-driven selection pressure, *i.e.* with *Best, Rank-based* or *Binary tournament* selection. On the other hand, without any selection pressure

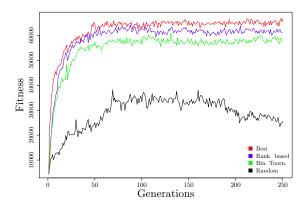


Figure 3: Median swarm fitness per generation over the 30 runs for the navigation task.

(*Random*), learning is much slower. Furthermore, for the three former selection methods the algorithm reaches comparable levels of performance in terms of median values of the swarm fitness. An exception can be noted for *Best* selection in the foraging task, which outperforms *Rank-based* and *Binary tournament*.

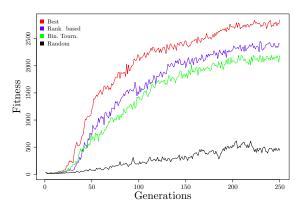


Figure 4: Median swarm fitness per generation over the 30 runs for the foraging task.

Despite the lower performances achieved by *Random*, the swarm still manages to learn behaviors for both tasks. This can be seen in the increasing trend of the median swarm fitness in Figure 3 and Figure 4. This result is expected on the navigation task. As it is the case in (Bredeche and Montanier (2010)), environmental pressure drives evolution toward behaviors that maximize mating opportunities and thus behaviors that explore the environment, increasing the swarm fitness.

The same trend is also observed on the foraging task. The improvement is slower but still present with *Random* selection. This could be explained by the fact that collecting

³Pairwise in this context means all combinations of pairs of selection methods, six combinations in our case.

items is a byproduct of maximizing mating opportunities. Agents collect items by chance while they navigate trying to mate. When inspecting the swarm in the simulator, we observed that, when selection pressure is present, the evolved behaviors drive the agents toward food items which means that the food sensors are in fact exploited. In other words, evolution drove the controllers to use these sensors. However, without any selection pressure (*Random*), there can not be a similar drive. We also observed this in the simulator: agents were not attracted by food items for *Random* selection.

When we analyze the comparison measures we introduced earlier, similar trends are observed. Figure 5 (respectively Figure 6) shows the box and whiskers plots of the four measures for each selection method over the 30 runs for the navigation task (respectively the foraging task).

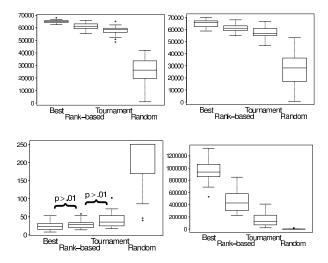


Figure 5: Box and whisker plots (30 independent runs) of the comparison measures for the four selection methods on the navigation task. From top to bottom and left to right: f_c , f_b , g_f and f_a . The label p>0.01 indicates no statistical difference for the corresponding two selection methods.

On the navigation task, the pairwise comparisons of the four measures, using Mann-Whitney tests at 99% confidence level, yield significant statistical difference between all selection methods except between *Best* and *Rank-based* (p-value=0.0795) and between *Rank-based* and *Binary tournament* (p-value 0.0116) in the case of the time to reach target (g_f) .

We also observe that Best reaches a higher swarm fitness for the fixed budget than the rest of selection methods, and this level is maintained at the end of evolution, as is shown in f_c and f_b (upper left and right in the figure). The target fitness level is rapidly reached for the three methods inducing selection pressure, and there is not significant difference between Best and Rank-based, nor between Rank-based and

Binary Tournament w.r.t g_f (lower left). Furthermore, in the case of Best, the required level is not only reached but surpassed during the entire evolution, leading to a value of f_a much higher than the ones of the rest of selection methods (lower right). However, this is not the case for Random selection that has much lower f_b (upper right) and f_c (upper left), and does not reach the target fitness level on more than half the runs that were launched (lower left and right).

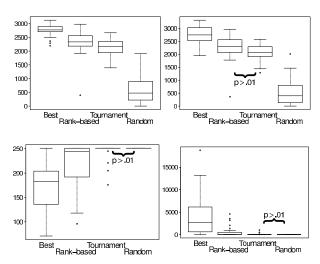


Figure 6: Box and whisker plots (30 independent runs) of the comparison measures for the four selection methods on the foraging task. From top to bottom and left to right: f_c , f_b , g_f and f_a . The label p>0.01 indicates no statistical difference for the corresponding two selection methods.

On the foraging task, there is a significant difference for all pairwise comparisons, except between Binary Tournament and Random in the case of the time to reach target, g_f , and the accumulated fitness above target, f_a , (p-value=0.0419 in both cases). This is explained by the fact that very few runs attained the target fitness⁴ in which case g_f is the last generation and f_a is almost zero. There is also no statistical difference between Rank-based and Binary Tournament on the fixed budget swarm fitness, f_b , (p-value=0.0105). This means that Binary Tournament reaches a fitness at the given budget that is comparable to the one of Rank-based, but it does not maintain this level, since for these two methods the difference is significant on f_c .

Best also gives better results on the foraging task: a high swarm fitness is reached and maintained at the end of evolution (f_b and f_c , upper left and right). It surpasses the target fitness level in almost all runs much faster and to a larger extent than Rank-based, that also manages to reach the required level for most runs (g_f , lower left), although by a

 $^{^4}$ For both tasks, the target fitness is 80% of the highest fitness reached by all methods during all runs.

lower level (f_a , lower right). This is not the case for *Tournament* and *Random* that do not achieve the target fitness level for most runs (lower left and right).

We can observe that all task-driven selection pressures yield much better performances on both tasks compared to *Random* selection. Consequently, we may conclude that selection pressure has a positive impact on performances, when solving a given task, and when the objective is not only to achieve adaptation of the swarm as it was the original motivation of mEDEA. Further, statistical tests show a direct correlation between the selection pressure and the performances achieved by the swarm on the two considered tasks. In other words, the stronger the selection pressure, the better the performances reached by the swarm.

In general, it has been argued that elitist strategies are not desirable in traditional EA's, and the same argument holds for traditional ER. This is due to the fact that elitist strategies may lead to a premature convergence at local optima. There exists an extensive body of work, especially in non-convex optimization, where it is preferable to explicitly maintain a certain level of diversity in the population to escape local optima and to deal with the exploration vs. exploitation dilemma. This requirement is perhaps not as strong in the context of distributed ER as our experiments show. Selection is performed among a portion of the population at the agent level, therefore, one might argue that these algorithms already maintain a certain level of diversity inherent to the fact that sub-populations are distributed on the different agents. Comparisons with other approaches in which separated sub-populations are evolved, such as spatially structured EA's (Tomassini (2005)) and island models (Alba and Tomassini (2002)), could give further insights on the dynamics of this kind of evolution.

Conclusions

In this paper, we studied the impact of task-driven selection pressures in on-line distributed ER for swarm behavior learning. This kind of algorithms raises several questions concerning the usefulness of selection pressure (partial views of population, noisy fitness values, etc.). We compared four selection methods inducing different intensities of selection pressure on two tasks: navigation with obstacle avoidance and collective foraging. Our experiments show that selection pressure largely improves performances, and that the intensity of the selection operator positively correlates with the performances of the swarm.

Foraging and navigation can be considered as relatively simple tasks and we believe that more complex and challenging ones, involving deceptive fitness functions, could give further insights on selection and evolution dynamics in the distributed case.

Acknowledgements

The authors would like to thank Evert Haasdijk for providing the version of RoboRobo simulator we used in the experiments, and for his insightful comments.

References

- Alba, E. and Tomassini, M. (2002). Parallelism and evolutionary algorithms. Evolutionary Computation, IEEE Transactions on, 6(5):443–462.
- Bianco, R. and Nolfi, S. (2004). Toward open-ended evolutionary robotics: evolving elementary robotic units able to self-assemble and self-reproduce. *Connection Science*, 16(4):227–248.
- Bredeche, N. and Montanier, J.-M. (2010). Environment-driven Embodied Evolution in a Population of Autonomous Agents. In *PPSN 2010*, pages 290–299, Krakow, Poland.
- Bredeche, N., Montanier, J.-M., Weel, B., and Haasdijk, E. (2013). Roborobo! a fast robot simulator for swarm and collective robotics. *CoRR*, abs/1304.2888.
- Dinu, C. M., Dimitrov, P., Weel, B., and Eiben, A. (2013). Self-adapting fitness evaluation times for on-line evolution of simulated robots. In *Proc. of GECCO'13*, pages 191–198. ACM.
- Eiben, A., Karafotias, G., and Haasdijk, E. (2010). Self-adaptive mutation in on-line, on-board evolutionary robotics. In Fourth IEEE Int. Conf. on Self-Adaptive and Self-Organizing Systems Workshop (SASOW), 2010, pages 147–152. IEEE.
- Eiben, A. E. and Smith, J. E. (2003). *Introduction to Evolutionary Computing*. Springer.
- Karafotias, G., Haasdijk, E., and Eiben, A. E. (2011). An algorithm for distributed on-line, on-board evolutionary robotics. In *Proc. of GECCO '11*, pages 171–178, New York, NY. ACM.
- Nolfi, S. and Floreano, D. (2000). Evolutionary Robotics: The Biology, Intelligence, and Technology. MIT Press, Cambridge, MA, USA.
- Noskov, N., Haasdijk, E., Weel, B., and Eiben, A. (2013). Monee: Using parental investment to combine open-ended and taskdriven evolution. In Esparcia-Alcázar, editor, *App. of Evol. Comput.*, volume 7835 of *LNCS*. Springer Berlin.
- Pineda, L., Eiben, A., and Steen, M. (2012). Evolving communication in robotic swarms using on-line, on-board, distributed evolutionary algorithms. In Chio, C. *et al.*, editor, *Applications of Evolutionary Computation*, volume 7248 of *LNCS*, pages 529–538. Springer Berlin Heidelberg.
- Silva, F., Urbano, P., Oliveira, S., and Christensen, A. L. (2012). odneat: an algorithm for distributed online, onboard evolution of robot behaviours. In *Artificial Life*, volume 13, pages 251–258. MIT Press.
- Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evol. Comput.*, 10(2):99–127.
- Tomassini, M. (2005). Spatially structured evolutionary algorithms. Springer Berlin.
- Watson, R. A., Ficici, S. G., and Pollack, J. B. (2002). Embodied evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, 39:1–18.

The Relay Chain: A Scalable Dynamic Communication link between an Exploratory Underwater Shoal and a Surface Vehicle

Becky Naylor¹, Mark Read², Jon Timmis¹ and Andy Tyrrell¹

¹ Department of Electronics, University of York

Abstract

In this paper we present the Relay Chain: a new algorithm central to a novel strategy for exploring underwater environments using a swarm of autonomous underwater vehicles (AUVs). The Relay Chain provides a mobile, scalable and dynamic communication link between the water's surface and a shoal of AUVs exploring the sea bed. The chain tracks the shoal as it explores, recruiting and returning AUVs from and to the shoal to modulate the length of the chain as required. The chain can instruct the shoal to reverse course when it travels too far from the starting point and there are no further AUVs to recruit. Given the challenging underwater environment, chain breakages are however inevitable, and as such we consider a number of recovery strategies to address chain breakages and minimise the time for which messages destined to the surface from the shoal are delayed. A simple 'turn and search' strategy is contrasted with strategies that depend on absolute positioning systems of varying accuracy. Implementing underwater absolute positioning systems is challenging, and our results highlight how accurate such a system must be to outperform more naive strategies, and hence be considered a worthwhile investment. We find the accuracy must be within 20cm, being 40% of AUV sensor range.

Introduction

Underwater exploration is a challenging task, the environment is noisy and difficult to access and navigate, and communication signals are short range due to heavy attenuation. The challenge can be met through a swarm of small autonomous underwater vehicles (AUVs) which divide aspects of the task between them and as a collective provide robustness in the difficult environment. The CoCoRo project is developing such a swarm of AUVs to address complex problems in a decentralised manner, sharing information only locally between robots (Schmickl T.et al. (2011)).

CoCoRo's underwater exploration strategy comprises two swarms: an *exploratory shoal* of AUVs that navigate the sea bed, and a *Relay Chain* of AUVs capable of relaying messages to one another, thus maintaining a communication link between the shoal and water's surface. We focus in this paper on the Relay Chain algorithm. The Relay Chain is a highly dynamic swarm: each link in the chain independently

coordinates its movement such that the chain tracks the shoal as it explores. The chain can recruit and return AUVs from and to the shoal into the chain to ensure it does not break when the shoal travels further way, and does not needlessly occupy AUVs when it returns. When there are no further AUVs that can be recruited, the chain can instruct the shoal to reverse course, further ensuring the chain does not break.

This paper presents the core Relay Chain algorithm, and an investigation into alternative strategies to deal with breakages in the chain when they occur. Despite the Relay Chain's dynamic and robust design, performing 3D collective coordination in noisy underwater environments using robots with limited sensor and communication ranges will certainly lead to AUVs in the chain losing contact with one another. We evaluate both a relatively naive recovery strategy whereby AUVs having lost contact with neighbouring links in the chain simply turn around and search for them, and a strategy based on AUVs localising one another through an absolute positioning system. Implementing an underwater absolute positioning system is highly challenging, and our analysis considers systems of varying accuracy, thereby highlighting how accurate such a positioning system must be to outperform the more simple strategy. The evaluation is based on the time required to send a message from the shoal to the water's surface, given that it may be delayed in an AUV that has temporarily lost contact with the next link in the chain. Recovery strategies must minimise the time for which the chain is broken, allowing the shoal to quickly report findings to the water's surface.

Both the Relay Chain algorithm design and the experiments reported here are performed in the CoCoRoSim simulation, which has been developed alongside, and calibrated to, the real-world CoCoRo AUV hardware in anticipation of its completion (Read M. *et al.* (2013)). Simulation represents a powerful tool for algorithmic development and evaluation, as sophisticated performance metrics can be applied with ease, and development can focus purely on algorithmic concerns free of hardware faults, and without the added time required to reprogram, deploy and collect AUVs. It has previously been employed in developing the exploratory shoal

² Charles Perkins Centre, The University of Sydney

algorithm used here (Read M. et al. (2013)).

Our manuscript is organised as follows. First we consider the CoCoRo AUV configurations and capabilities, and the simulation environment. We summarise the previously published shoaling algorithm, used here by the exploratory shoal, before describing the Relay Chain algorithm itself. Thereafter we report our hypothesis-driven experiments to evaluate potential chain-breakage recovery strategies, and then conclude the paper.

AUV Configuration and Simulation

The CoCoRoSim Netlogo3D simulation reflects the design of the bespoke AUV hardware developed for CoCoRo (these AUVs are named "Jeff"), and has been calibrated to provide representative behaviour of the real AUVs. The Jeff platform is 25 x 12 x 5cm in dimension, uses thrusters to propel itself forwards or backwards and control its yaw, and can adjust its depth through a buoyancy pump. Omnidirectional communication with a range of 50cm is provided by a radio-frequency (RF) modulator. Six bluelight sensor systems, comprising LEDs and photodiodes, located on the front, back, left, right, top and bottom of the AUV provide detection of obstacles and other AUVs. Each sensor system can perceive the distance to obstacles within an angle of 120 degrees and a range of 50cm; exact triangulation within this cone is not possible, only distance can be discerned (Scuola Superiore Sant'Anna and CoCoRo partners (2012); Universit libre de Bruxelles and CoCoRo partners (2013)).

The CoCoRoSim Netlogo3D simulation reflects sensor and actuator capabilities of Jeff, and its physics engine has been calibrated against empirical data generated from the previous CoCoRo AUV design (Read M. et al. (2013)). The water in the simulator causes drag for translational and rotational movement. The robot arena (Figure 5) is a rectangular tank with a fixed 'water level' as the top surface, the walls and floor of the tank are represented as solid surfaces which are detectable by the bluelight sensors.

Exploratory Shoal

The exploratory shoal uses an algorithm based on Reynolds' boids (Reynolds, 1987), the implementation of which is described in Read M. et al. (2013); only a brief summary is provided here. An AUV's movement is dictated by the weighted sum of three desired trajectories, calculated based on rules for cohesion, alignment and separation. Separation provides a trajectory away from other AUVs found within a threshold distance, and is designed to prevent collisions. The cohesion trajectory points towards other AUVs within detection range, and prevents AUVs losing contact with one another. The alignment trajectory reflects the average orientation of neighbouring AUVs, and seeks to align AUV orientations. The trajectories for each rule are calculated at regular time intervals, and their weighted sum governs AUV thruster and buoyancy pump actuation.

Separation and cohesion rules are informed by distance data provided by the bluelight sensors, with distance measures beyond 95% of the maximum sensor range excluded to ignore sensor noise. The separation threshold is 30% of sensor maximum range. The alignment rule uses data obtained from AUVs sharing their heading information with one another over RF.

Relay Chain Algorithm

The Relay Chain algorithm controls the initial formation and the maintenance of a communication link between the exploratory shoal and the surface.

Formation

A group of AUVs is deployed on the water surface, some of these will form the Relay Chain and the rest will join the exploratory shoal. The process is shown in Figure 1. All of the AUVs begin the process in the *explore* state (Figure 1). One AUV is chosen to be the 'seed' to start the formation. This becomes the start chain AUV (Figure 1). The start AUV remains stationary in the centre at the surface of the water. It is representative of any surface vehicle such as a fixed base station or a boat. The algorithm can also work with a mobile surface vehicle, with the chain following the vehicle in the same manner as the exploratory shoal, but for simplicity it remains stationary in the present experiments. This start AUV requests a number of vehicles to go to specific depths, each represents one layer of the chain. These will be the middle chain AUVs and the last one the end chain AUV (Figure 1).

For example in a water depth of 1m, five AUVs are recruited by the seed to initialise the chain. The remaining vehicles follow the chain downwards and form the exploratory shoal. The AUVs head to a target depth calculated using a user defined distance between AUVs. Once in position each AUV in the chain communicates with the AUVs either side of it, to keep at a depth halfway between the two of them. Each AUV in the chain knows the ID number of the 'next' and 'previous' AUVs in the chain, this is labelled from top to bottom so next is the one below and previous is the one above it. So that the algorithm remains decentralised each AUV is only aware of its local position and does not know globally where it is within the chain. An image of chain and shoal after initialisation is shown in Figure 2, chain AUVs are black and shoal AUVs are white.

Chain States

The Relay Chain algorithm splits the AUVs into task driven subgroups of: the exploratory shoal, the Relay Chain and navigation. These are the highest level states *explore*, *chain* and *navigate* in the state diagram (Figure 1).

Chain AUVs maintain a *position* state, to identify whether they are the *start* AUV at the top of the chain, a *middle* AUV or the *end* AUV at the bottom of the chain. The start and

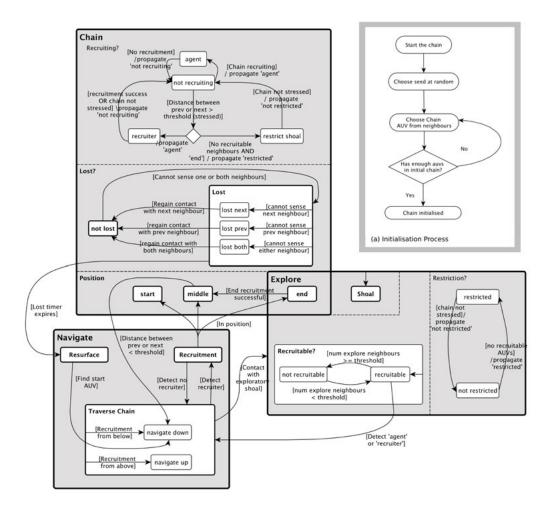


Figure 1: Complete state diagram illustrating the operation of the Relay Chain algorithm. The three core states are *chain* AUVs in the Relay Chain, *explore* AUVs in the exploratory shoal and *navigate* AUVs moving between *chain* and *explore*. The initialisation flow chart is shown in (a)

end AUVs are special cases. As discussed in the initialising section the start AUV remains at the water's surface and initialises chain formation. The *end* AUV is both a member of the chain and the exploratory shoal (see Figure 1). This means that it follows the shoal as it explores the environment and provides a link to the rest of the chain.

To maintain contact with the exploratory shoal the chain can dynamically change length by recruiting more AUVs or having AUVs leave the chain. The chain must add more AUVs to extend if the shoal moves away from the start AUV, so the chain has 'too few' members. It can also gain 'too many' members, for example if the shoal changes direction and returns towards the chain. Whether the chain has 'too many' or 'too few' AUVs is determined by thresholding the distance between chain AUVs.

To determine their desired position all *middle chain* AUVs monitor the distance and heading to the AUVs on either side

of them in the chain. They aim to stay in a position halfway between the AUVs on either side of them. If the halfway distance exceeds a 'stress threshold' then it will stay nearer the 'previous' neighbour (one above in the chain). This is so that the 'stress' is transferred down the chain, nearer to the exploratory shoal; making it easier for AUVs to be recruited directly into the chain.

When the distance between the shoal and *end* chain AUV reaches a threshold the *end* AUV in the chain switches to the *recruiter* state (in the *Recruiting?* section in *chain* Figure 1). It propagates a message through the other chain AUVs to become *agents* to direct any *recruitable* AUVs to the site of the *recruiter*. The *recruitable* AUV will enter either *navigate up* or *navigate down* state depending on the direction it is given by the 'agent'.

A *recruitable* AUV is typically one from the exploratory shoal (*explore* state in Figure 1), but an AUV that is already

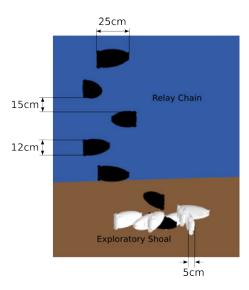


Figure 2: CoCoRo simulation environment, Relay Chain robots are black and exploratory shoal are white.

traversing the chain can also be recruited. If a recruitable AUV senses a recruiter or agent it enters the traverse chain state. Once it finds a recruiter the AUV will enter the recruitment state and manoeuvre into position to join the chain (as shown in the navigate state in Figure 1). All available AUVs will respond to a recruiter, but once the first responder is in position it joins the chain and the recruitment process stops (unless more AUVs are required). When a recruitment AUV can no longer see a recruiter it returns to the traverse chain state. When a traverse chain AUV cannot see a recruiter or agent and has contact with the exploratory shoal then it returns to the explore state. Once recruited, the new AUV will become the end AUV and the original end AUV will change state to become a middle AUV. (Position section in chain state Figure 1.

Once *recruitment* is successful, or if the distance between the end AUV and its previous neighbour has reduced below the 'stress threshold' by movement of the shoal, the *end* chain AUV switches to *not recruiting*. It will then propagate this message to the other chain AUVs to switch to the *not recruiting* state from the *agent* state (as seen in the *recruiting*? section in *chain* Figure 1).

If there are insufficient *recruitable* AUVs within range of the *end* chain AUV then the *end* AUV will pass a *restrict* message to the shoal. The number of *recruitable* AUVs required in the shoal is determined by a user specified threshold, its setting depends on the number of AUVs being used. Here for example with 15 AUVs in the simulation the 'recuritable threshold' is 2. The *end* AUV transmits a message to the shoal AUVs in range to enter the *restricted* state. They propagate this to other shoal members, outside of the range of the *end* chain AUV (in *restriction?* section in the *explore* state in Figure 1). The entire shoal is either *restricted* or

not restricted, though individual AUVs may have opposing states while the message propagates through the shoal. Once in the restricted state the exploratory shoal AUVs will turn around (180 degrees) and head back towards the chain. This reduces the distance between the chain members and minimises the risk of the chain breaking.

To identify whether there are unnecessary AUVs in the chain each of the *middle* position chain AUVs compares the distance between themself and the AUVs either side of them in the chain to a 'slackness threshold'. Both this and the 'stress threshold' are a proportion of the bluelight range of the AUVs. If an AUV is too close to its neighbour, according to the threshold, the AUV will send an RF message to the next and previous AUVs in the chain announcing that it is leaving. The next and previous IDs of both neighbour AUVs are updated to remove the AUV from the chain. It will then enter the *traverse chain navigate* state and *navigate down* to the exploratory shoal as shown in Figure 1. Once it has contact with the exploratory shoal it changes to the *explore* state to join the shoal. The *traverse chain* AUVs move alongside the chain, as shown in Figure 3.

As shown in Figure 1, all *chain* AUVs have a *lost* status. A *chain* AUV is considered *lost* if it loses bluelight contact with either of its neighbours. This is further split to identify on which side the AUV has lost contact with. If an AUV remains *lost* for a given duration (e.g. 1 minute) then it is told to *resurface*. AUVs in the *resurface* state head to the surface at the coordinates they are currently at. They then use GPS to find the *start* AUV at the surface of the water, then enter the *traverse chain* state to rejoin the exploratory shoal or be recruited into the chain.

All of the AUVs utilise shoaling behaviours as described in the Exploratory Shoal section. The parameters are set differently depending on the desired behaviour, by weighting each of the rules' importance. For example; in the unrestricted *explore* state AUVs have a balance between cohesion, alignment and separation to produce shoaling behaviour. When in the *traverse chain* state the separation rule is most important, so that the AUV keeps moving rather than being attracted to a chain AUV and staying on one of the layers. The cohesion has a weight half that of the separation, so that the AUV follows the chain and does not head off in some other direction. The alignment rule is given a weight of 0, because the AUV is not in a shoal to align with.

Recovering lost AUVs

Knowledge of position is important for the Relay Chain algorithm, because the AUVs aim to maintain a position based on their neighbours. Using only the onboard sensors it is possible to work out the necessary heading locally. An AUV triangulates where the neighbour is by detecting which of its bluelight sensors can sense the neighbour. A problem arises when an AUV moves out of bluelight range of one or both of its neighbours. It no longer has any positional information

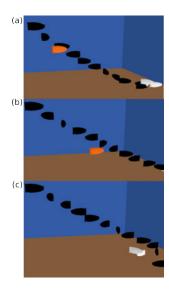


Figure 3: An AUV traversing down the chain to the exploratory shoal, the traversing AUV is shown in orange. Stage (a) shows it leaving the chain after the distance between it and its neighbours reduced below a threshold. Stage (b) shows the progress down the chain and stage (c) shows it joining the exploratory shoal.

about the neighbour(s). Such an AUV is considered 'lost' and each AUV has a lost status to store which neighbours (above or below) it has lost.

If one or more AUVs are lost, communications can no longer be passed along the chain. To solve the problem of finding the chain when lost, two alternative strategies are proposed and compared.

The first strategy is called 'Turn and Search', it is the simpler of the two strategies and can be performed without any additional hardware. In Turn and Search, when an AUV becomes lost it turns around by 180 degrees with a random offset of \pm 5 degrees (chosen using a uniform distribution). A 100s timer is started, again with an added uniformly chosen random component of up to 150s. If the timer expires and the AUV has not refound the chain then it turns again. This process continues until the chain is found. An example path is shown by the trace in Figure 4. The AUVs still use their shoaling parameters and avoid collisions.

In Turn and Search recovery, if the AUV remains lost for over a threshold time of 400s then it resurfaces to find the chain from the top. The threshold can be adjusted depending on the desired task and importance of refinding the chain.

The second strategy is Absolute Positioning (AP). While this may generally seem a more suitable positioning solution than the use of bluelight triangulation, the accuracy of such systems is limited. As the AUVs being used are small the accuracy of the position is important. This strategy requires additional hardware, so is only desirable if Turn

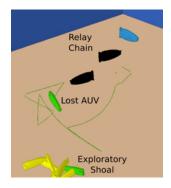


Figure 4: Example path taken by a lost AUV using Turn and Search recovery. The green AUV (fourth from the top) is lost and the trail is the path it has taken.

and Search is ineffective.

Positioning underwater cannot be achieved with GPS but there are different solutions including hydro-acoustic, image based and inertial systems. Of these, acoustic systems are most applicable to CoCoRo AUVs and tank based test scenarios. Image based systems use differences in the environment surrounding the AUVs, for example the seabed (Garcia et al. (2001)). This is not well suited to the tank environment used for testing the CoCoRo AUVs. Inertial systems are unlikely to offer any advantage over the existing system. They use measurements from gyroscopes and accelerometers on the AUV, so errors accumulate (Garcia et al. (2004)).

Acoustic systems use a set of beacons with fixed positions around the area in which the AUVs operate. The AUVs communicate acoustically with these beacons and the time for the signal to arrive is used to calculate the position (Alcocer et al. (2006)). The beacons can be attached to buoys with GPS receivers, so that they know their absolute position.

To provide specifications for a recovery system we compare the success of each method. We test three hypotheses:

- 'Using AP rather than Turn and Search recovery will reduce the mean time to wait before a message can be sent along the chain.'
- 2. 'The higher the accuracy of the AP the lower the average time to wait before a message can be sent along the chain'.
- 3. 'As the accuracy of the AP is reduced there will be a point when using AP only when an AUV is lost outperforms using AP all of the time'

To test these hypotheses we run two experiments. In the first Absolute Positioning (AP) is used to control the AUVs all of the time and in the second a combined bluelight triangulation and AP method. In experiment 2 the AP is only used when the AUV is lost.

In both experiments the results are compared to bluelight triangulation with Turn and Search recovery and to a control using 'random' movement.

Method

The simulator maintains absolute position information for each AUV, so when lost the AUVs can head back directly to their neighbour. In reality underwater systems will not have this type of sub-centimetre accuracy. To achieve this discrepancy in accuracy an offset is added to the absolute position value of both the lost AUV and the neighbour that it is heading back towards. This offset is randomly chosen from the specified range to each of the x, y and z coordinates. The value is chosen with uniform probability because this is the worst case scenario. This means that the AUV knows the position of its lost neighbour (or neighbours) within a cube centred on its real position. A different offset is used for the AUV's own value and each of its neighbours' values. Each time the AUV becomes lost a different offset is generated. As the AUVs have a depth sensor accurate to 0.1mm no offset is added to the lost AUV's own depth coordinate.

The experiment is run in a tank with a width and depth of 5m and a water level of 1m. All AUVs begin at the water level, in the centre of the tank with randomised starting headings. There are 15 AUVs, of which 5 (including the seed) are set to specific depths as described in the initialisation section.

The seed is fixed in the centre of the tank at the water surface. The 4 other chain AUVs are initialised to depths 15cm apart. The rest of the AUVs enter the 'explore' state and head down the chain. They are either recruited or form the exploratory shoal. At least 2 of the AUVs must be in the *explore* state. This provides enough length in the chain to extend to each corner of the tank when exploring. An example of this can be seen in Figure 5.

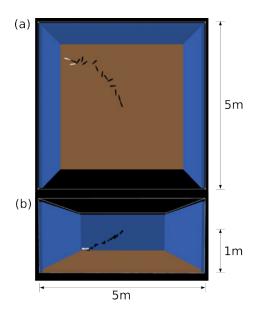


Figure 5: The Relay Chain in the tank used for experiments (a) top down view (b) front view.

In experiment 1 'partial AP' the AUVs will use bluelight triangulation when they are in range and AP when they are lost. For experiment 2 'full AP' the AUVs will always use AP to hold their position in the chain. Although they do not use bluelight triangulation the AUVs can still be 'lost' because once out of bluelight range they are out of communication range.

The simulation is run for 1 hour in simulation time. 1hr is sufficiently long for the shoal to explore the bottom of the tank. The simulation is repeated 500 times for each of the different AP accuracies. Offsets of between 0cm and 50cm are tested, 50cm represents a substantial offset as the AUVs are only 25cm by 12cm by 5cm. The range of accuracies is compared to the system without any absolute positioning, using the Turn and Search recovery method and also to a control version with 'random walk' movement with all AUVs in the Turn and Search recovery state at all times. The AUVs are given an initialisation period of 1 minute to allow them to reach their desired depths, then enter the recovery state for the remainder of the simulation run.

The success of the system is measured using the average time that it takes to send a message from one end of the chain to the other. This is important because the chain is necessary to convey messages, if the message takes too long then the information contained may no longer be valid.

The time between sending and receiving comprises two parts; firstly the time to physically transmit the data along the chain and secondly the time to wait until there are no lost AUVs so all AUVs can communicate with their neighbours.

The time to send the message itself is small compared to the potential time to wait until there are no lost AUVs. A typical message is 50 bytes and the AUVs can transmit at 28.8Kb/s. Each transfer will take 0.0139s and the chain can need no more than 13 transfers from one end to the other. Assuming negligible processing time the entire process will take at most 0.181s. The likelihood of the chain breaking while a message is physically being propagated is small, and the message can be resent if this occurs. The 'time to wait' component is therefore the critical factor, rather than the 'transmit time'.

The waiting time is determined by the contiguous time that the chain is broken for. For example if the chain is broken for a total duration of 60s then it is much preferable to have six 10s blocks than one 60s block. The chain is able to send a message in the gaps between the 10s blocks, rather than waiting the entire 60s before sending a message.

Results

The results using full AP are shown in Figure 6. In the graph 0 (no offset) is most accurate, each subsequent value is the offset in cm. The first column shows the control of 'random' movement when all AUVs are in Turn and Search recovery. The second column shows the system with no positional information other than the local bluelight triangulation.

The A test (Vargha A. *et al.* (2000)) is a non-parametric effect magnitude test which allows the comparison of the distribution of results for a comparison parameter with the distributions using alternative values. The A test indicates the probability that a randomly chosen sample from one distribution is larger than a randomly chosen sample from the other. A value of between 0 and 1 is returned by the test, with values higher than 0.71 or lower than 0.29 indicating 'large' differences between distributions. Large differences are assumed to be significant (Vargha A. *et al.* (2000)).

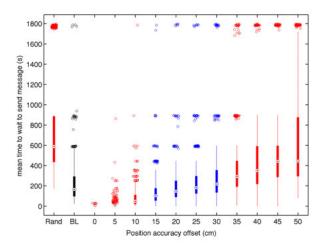


Figure 6: Mean time to wait before a message can be sent using full AP. Rand is random movement, BL is bluelight with Turn and Search (this is the A test comparison distribution) and numerical values are the AP offset.

The A test responses using different comparison values are provided in table 1. They show whether the difference between each column is significant. The colour of the column in Figure 6 represents the A test score of each distribution with a comparison to bluelight with Turn and Search (black in the graph). 'Red' columns are significantly different to the comparison distribution, 'blue' columns are not (values for each column are given in table 1).

From these the Turn and Search recovery method is significantly better than the random control. AP is significantly better than Turn and Search with an offset of up to 10cm, but significantly worse with an offset of 35cm or greater. An offset of 40cm or greater is not significantly different to the random control. When considering how much offset it takes before the system does not perform as well, 5cm is not significantly different to having no offset.

The results for partial AP (experiment 2) are shown in Figure 7. The A test responses are shown in table 1. For the partial AP, unlike full AP, all offsets are significantly better than the random control. Comparing the BL column with the AP columns shows that the AP values between 0 and 20 are

significantly better than the bluelight with Turn and Search. For partial AP offsets of 5 and 10cm are not significantly different to having no offset.

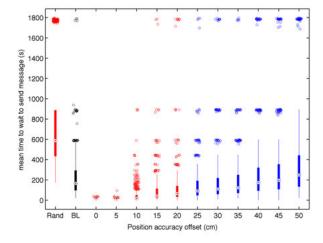


Figure 7: Mean time to wait before a message can be sent using partial AP. Rand is random movement, BL is bluelight with Turn and Search (this is the A test comparison distribution) and numerical values are the AP offset.

Discussion

The results presented above provide responses to the three hypotheses presented earlier. *Hypothesis 1 'Using AP rather than Turn and Search recovery will reduce the mean time to wait before a message can be sent along the chain.'* is shown to be true for both full and partial AP when the offset is 10cm or lower. The median average time to send a message (of the 500 runs) is 17.8s for full AP with 0 offset and 17.2s with partial AP compared to 166.4s with Turn and Search.

With too high an offset AP performs no better (or in the full AP experiment worse) than Turn and Search. This means for the hypothesis to be correct, the chosen AP system must be able to deliver accuracy to within 10cm for full AP and 20cm for partial AP.

Hypothesis 2 'The higher the accuracy of the AP the lower the average time to wait before a message can be sent along the chain' is also shown to be correct. It is true for both full and partial AP. There is a marked reduction in performance as the accuracy is decreased.

The system can function without significant degradation in the time to send a message, with an offset of 10cm when compared to a 0cm offset. When compared with a bluelight only system, an offset of up to 20cm is shown to be preferable. From this we can specify a system with an accuracy of 20cm or better be used.

Hypothesis 3 is 'As the accuracy of the AP is reduced there will be a point when using AP only when an AUV is

Exp	Comparison	A test result												
		Rand	BL	0	5	10	15	20	25	30	35	40	45	50
Full	Rand	0.50*	0.10	0.00	0.00	0.02	0.05	0.08	0.13	0.16	0.24	0.29	0.38	0.41
AP	BL	0.90	0.50*	0.00	0.03	0.19	0.33	0.46	0.57	0.61	0.72	0.76	0.83	0.84
1hr	AP 0	1.00	1.00	0.50*	0.33	0.81	0.98	0.99	0.99	1.00	1.00	1.00	1.00	1.00
Part	Rand	0.50*	0.10	0.00	0.00	0.01	0.03	0.04	0.05	0.09	0.09	0.15	0.16	0.21
AP	BL	0.90	0.50*	0.00	0.00	0.09	0.19	0.24	0.32	0.38	0.42	0.51	0.56	0.63
1hr	AP 0	1.00	1.00	0.50*	0.35	0.61	0.83	0.94	0.98	0.99	1.00	0.99	0.99	1.00

Table 1: The A test results for wait time experiments with different parameters set as the comparison distribution, here marked with *. Values higher than 0.71 or lower than 0.29 are significantly different (in red).

lost outperforms using AP all of the time'. Comparing the partial and full AP results shows that for values of 0, 5 and 10 there is no significant difference. For values of 15 and above the partial system is significantly better than the full system. An offset value of 15cm represents the switching point in performance. This result can be explained because with full AP the positioning is always altered by the offset. In partial AP the AUVs are able to use their bluelight sensors to triangulate their relative position, so are not affected by the offset at this time.

The clusters in Figures 6 and 7 can be explained by AUVs getting lost and not finding the chain again. The highest valued cluster is at the duration of the run.

In summary, for AP to be worthwhile for recovery of lost AUVs to the Relay Chain it must be accurate to within 20cm otherwise bluelight triangulation is preferable.

Conclusion

The Relay Chain has been shown to be a decentralised method of providing a link between AUVs exploring the environment and the surface of the water. It is scalable, resizing as the shoal moves around the environment.

Restriction of the shoal's movement if there are not enough recruitable AUVs prevents the Relay Chain breaking and keeps the shoal in communication range of the surface vehicle. Messages can be propagated in either direction along the chain, relaying exploration data from shoal to surface or instructions to the shoal.

A disadvantage of the Relay Chain is that the number of AUVs used restricts the number available in the shoal. This is a result of limited communication range underwater.

Extensions where the chain is likely to be successful include fault tolerance, because the decentralised nature of the algorithm means an AUV can be removed and the chain reposition and continue operating. The introduction of a more challenging environment with low visibility water should be handled by the same recovery mechanisms used for lost AUVs in clear water.

The challenges of underwater communication and positioning mean that the CoCoRo Jeff AUVs can lose sight of their neighbours in the chain. Once lost, using only the available sensors it is difficult to refind the chain. Two alternative recovery strategies are used to overcome this. Either method

provides some robustness to lost AUVs, though Absolute Positioning gives the best performance and is preferable to bluelight when the accuracy is within 20cm.

Acknowledgements

We thank Christoph Möeslinger and colleagues at the University of Graz for their contribution to CoCoRoSim. We also thank all other members of the CoCoRo Consortium.

This work is supported by the EC funded CoCoRo Project, GA 270382. JT is also part funded by the Royal Society and the Royal Academy of Engineering.

References

- Alcocer, A., Oliveira, P., and Pascoal, A. (2006). Underwater acoustic positioning systems based on buoys with gps. In *Proceedings of the Eighth European Conference on Underwater Acoustics*, volume 8, pages 1–8.
- Garcia, J.-E., Kyamakya, K., and Jobmann, K. (2004). Positioning issues in underwater acoustic sensor networks. In WPNC 2004, pages 129–136.
- Garcia, R., Batlle, J., Cufi, X., and Amat, J. (2001). Positioning an underwater vehicle through image mosaicking. In *Robotics* and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on, volume 3, pages 2779–2784.
- Read M. *et al.* (2013). Profiling underwater swarm robotic shoaling performance using simulation. *TAROS 2013*, pages 456–462.
- Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21(4):25–34.
- Schmickl T.et al. (2011). Cocoro the self-aware underwater swarm. In SASOW 2011 Fifth IEEE Conference on, pages 120–126.
- Scuola Superiore Sant' Anna and CoCoRo partners (2012). Propulsion mechanism for swarm swimming and buoyancy-elastic mechanism. *CoCoRo Deliverable 1.2*.
- Universit libre de Bruxelles and CoCoRo partners (2013). Report or publications on collective measures of the environment. *CoCoRo Deliverable 5.2*.
- Vargha A. et al. (2000). A critique and improvement of the cl common language effect size statistics of McGraw and Wong. J. of Edu. and Behav. Stats., 25(2):101–132.

Collective Behaviors

Scale-free correlations in collective motion with position-based interactions

E. Ferrante¹, A. E. Turgut¹, T. Wenseleers¹, and C. Huepe²

¹Laboratory for Entomology, Katholieke Universiteit Leuven, 59 Naamsestraat - bus 2466, 3000 Leuven, Belgium ²Northwestern Institute on Complex Systems, Northwestern University, Evanston, IL 60208, USA and CHuepe Labs, 954 W. 18th Place, Chicago IL 60608, USA

Extended Abstract

Introduction Collective Motion (CM) is observed in a variety of animal groups such as bird flocks and fish schools. In a recent study, Cavagna et al. (2010) found that the correlation lengths of speed and velocity fluctuations in starling flocks are not set by a specific interaction range, but are instead scale-free, proportional to the group size. So far, this observation has been justified by hypothesizing that flocks evolved to follow critical dynamics near a phase transition. where scale-free correlations are known to emerge. Criticality could provide an evolutionary advantage by allowing the flock to optimally respond to an external perturbation such as a predator attack. However, a criticality-based explanation may only be required in cases where interactions are based exclusively on relative orientations, as often assumed in CM models, following the seminal work by Vicsek et al. (1995). In this paper, we show that an alternative, more parsimonious, mechanism can produce scale-free correlations when considering interactions based on relative positions.

AE simulations We consider an active elasticity (AE) model (Ferrante et al., 2013), where N self-propelled agents are moving in 2D, with neighbors permanently linked by spring-like linear forces. Given the position \vec{x}_i and orientation θ_i of each agent i, the AE model is defined as:

$$\dot{\vec{x}}_i = v_0 \, \hat{n}_i + \alpha \, \left(\vec{F}_i \cdot \hat{n}_i \right) \, \hat{n}_i, \tag{1}$$

$$\dot{\theta}_i = \beta \left(\vec{F}_i \cdot \hat{n}_i^{\perp} \right) + D_{\theta} \, \xi_{\theta}.$$
 (2)

Here, v_0 is the preferred self-propulsion speed, α and β are the speed and angular force coupling coefficients, and \hat{n}_i and \hat{n}_i^{\perp} are unit vectors pointing parallel and perpendicular to the heading of agent i, respectively. The sum of elastic forces over agent i is given by: $\vec{F}_i = \sum_{j \in S_i} \left(-k/l_{ij}\right) \left(\|\vec{r}_{ij}\| - l_{ij}\right) \left(\vec{r}_{ij}/\|\vec{r}_{ij}\|\right)$, where S_i is the set of all j neighbors linked to agent i, the spring constants k/l_{ij} and natural lengths l_{ij} characterize the interactions between them, and $\vec{r}_{ij} = \vec{x}_j - \vec{x}_i$ are their relative positions. Noise is introduced by adding $D_{\theta} \, \xi_{\theta}$ in Eq. (2), where D_{θ} is the noise strength and ξ_{θ} a random variable with standard, zero-centered normal distribution of variance 1.

Simulations were carried out by integrating Eqs. (1) and (2) numerically. They were set up as a minimal AE-based version of the starling experiments by Cavagna et al. (2010). Parameters $l_{ij}=1$, $v_0=10$, k=100, $\alpha=2$, and $\beta=3$ were chosen to roughly mimic the experimental dynamics. The noise level was set to be high $(D_{\theta} = 0.628)$ but far from any critical point (here at $D_{\theta}^* \approx 1.1$, where the system transitions to a disordered, non-flocking state). We focus on the correlations that develop as the group turns due to strong local perturbations. This is continuously occurring in natural flocks as individuals at the edge of the group change their heading directions based on external stimuli. We mimic such situation by initializing all agents with $\theta = 0$, except for one (the *informed* agent), which is forced to head towards $\theta =$ $\pi/3$ during the whole run, while its speed is determined by Eq. (1). This corresponds roughly the natural flock turning dynamics described in (Attanasi et al., 2013).

Figure 1 displays snapshots of the AE simulations. At t=0.5 (a), the top-right corner of the hexagon is turning towards $\theta=\pi/3$, as imposed by the informed agent, while the rest continues heading to the right, still unperturbed. In the subsequent t=1.0 (b), 1.5 (d), and 2.0 (e) snapshots, the whole group starts to acquire the $\pi/3$ orientation, as the local perturbation spreads out through the system. Despite the turning and imposed noise, the degree of agent alignment remains high, with polarization order parameter $\psi=\|\sum_{i=1}^N \vec{v}_i\|/(Nv_0)$ ranging between 0.90 and 0.96 ($\psi=1$ indicates full alignment), which is consistent with the starling experiments. All panels display the informed agent detached from the hexagon, above and to the left of its top-right corner, since its virtual springs sustain large forces as they induce group turning, and are therefore very stretched.

Correlation function analysis We focus on the speed fluctuations $s_j = \|\vec{v}_j\| - \frac{1}{N} \sum_{i=1}^N \|\vec{v}_i\|$ and velocity fluctuations $\vec{u}_j = \vec{v}_j - \frac{1}{N} \sum_{i=1}^N \vec{v}_i$. As in (Cavagna et al., 2010), we compute the correlation function for the latter using

$$C_{\vec{u}}(r) = K \frac{\sum_{i,j=1}^{N} \vec{u}_i \cdot \vec{u}_j \, \delta(r - r_{ij})}{\sum_{i,j=1}^{N} \delta(r - r_{ij})},$$
 (3)

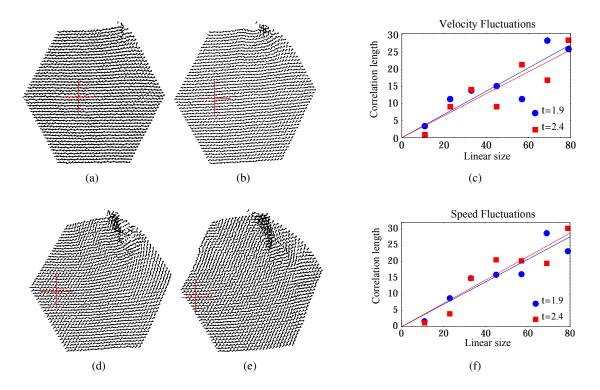


Figure 1: Simulation snapshots at t = 0.5 (a), 1.0 (b), 1.5 (d), and 2.0 (e). Red crosses indicate where the hexagon's center was at t = 0. Correlation lengths of the velocity (c) and speed fluctuations (f) as a function of group sizes at two different times.

and the correlation function $C_s(r)$ for the speed fluctuations using an equivalent expression, but where \vec{u}_i is replaced by s_i and the dot product by multiplication. Here, r_{ij} is the distance between agents i and j, $\delta(\cdot)$ is a smoothed Dirac delta function, and K is a normalizing constant defined so that $C_{\vec{u}}(0)=1$. Both functions must cross zero since s_i and \vec{u}_i have zero mean. We thus define as correlation lengths the first zero-crossing points of $C_{\vec{u}}(r)$ and $C_s(r)$.

We performed the simulations and analysis for hexagonal groups containing $N=91,\ 397,\ 817,\ 1519,\ 2437,\ 3571,$ and 4681 agents, a range slightly larger than in the starling experiments. Correlation functions were computed at t=1.9 and 2.4 (two somewhat arbitrary instants, chosen slightly after the perturbation crosses our largest hexagon) in order to examine the typical correlation lengths and their degree of variation. Panels (c) and (f) display the correlation lengths of the velocity and speed fluctuations, respectively, as a function of the linear size of the group, defined here as the distance between opposing vertices of the undeformed hexagon. While we observed significant variations throughout the dynamics, both quantities are shown to be large and proportional to the group size, as in Cavagna et al. (2010).

Discussion and conclusion The results above show that the scale-free correlations observed in starling experiments are not necessarily due to a critical regime, but can be a natural consequence of position-based interactions among in-

dividuals. Our work also provides a bio-inspired algorithm that can produce coherent, group-level collective motion for robot swarms or other artificial flocks. By implementing the AE model, highly correlated motion dynamics that always span a significant and approximately constant fraction of the group, regardless of its size, can be achieved.

Acknowledgements This work was partially supported by the Vlaanderen Research Foundation Flanders (H2Swarm project), the US National Science Foundation (Grant No. PHY-0848755) and TUBITAK (Grant No. 2219).

References

Attanasi, A., Cavagna, A., Del Castello, L., Giardina, I., Grigera, T. S., Jelic, A., Melillo, S., Parisi, L., Pohl, O., Shen, E., and Viale, M. (2013). Superfluid transport of information in turning flocks of starlings. *Cond-mat.stat-mech*, arXiv:1303.7097.

Cavagna, A., Cimarelli, A., Giardina, I., Parisi, G., Santagati, R., Stefanini, F., and Viale, M. (2010). Scale-free correlations in starling flocks. *Proceedings of the National Academy of Sciences*, 107:11865–11870.

Ferrante, E., Turgut, A. E., Dorigo, M., and Huepe, C. (2013). Collective motion dynamics of active solids and active crystals. *New Journal of Physics*, 15(9):095011.

Vicsek, T., Czirok, A., Ben-Jacob, E., Cohen, I., and Shochet, O. (1995). Novel type of phase transition in a system of selfdriven particles. *Physical Review Letters*, 75(6):1226.

Asynchronous Evolution: Emergence of Signal-Based Swarming

Olaf Witkowski and Takashi Ikegami

University of Tokyo, Japan olaf@sacral.c.u-tokyo.ac.jp

Abstract

Since Reynolds boids, swarming behavior has often been reproduced in artificial models, but the conditions leading to its emergence are still subject to research, with candidates ranging from obstacle avoidance to virtual leaders. In this paper, we present a multi-agent model in which individuals develop swarming using only their ability to listen to each others signals. Our model uses an original asynchronous genetic algorithm to evolve a population of agents controlled by artificial neural networks, looking for an invisible resource in a 3D environment. The results demonstrate that agents use the information exchanged between them via signaling to form temporary leader-follower relations allowing them to flock together.

Introduction

The ability of fish schools, insect swarms or starling murmurations (Figure 1) to shift shape as one and coordinate their motion in space has been studied extensively because of their implications for the evolution of social cognition, collective animal behavior and artificial life (Couzin 2009).



Figure 1: Starling murmuration¹

Swarming is the phenomenon in which a large number of individuals organize into a coordinated motion. Using only the information at their disposition in the environment, they are able to aggregate together, move *en masse* or migrate

towards a common direction.

The movement itself may differ from species to species. For example, fish and insects swarm in three dimensions, whereas herds of sheep move only in two dimensions. Moreover, the collective motion can have quite diverse dynamics. While birds migrate in relatively ordered formations with constant velocity, fish schools change directions by aligning rapidly and keeping their distances, and insects swarms move in a messy and random-looking way (Budrene et al. 1991, Czirók et al. 1997, Shimoyama et al. 1996).

Numerous evolutionary hypotheses have been proposed to explain swarming behavior across species. These include more efficient mating, good environment for learning, combined search for food resources, and reducing risks of predation (Zaera et al., 1996). Partridge and Pitcher (1979) also mention energy saving in fish schools by reducing drag.

In an effort to test the multiple theories, the past decades counted several experiments involving real animals, either inside an experimental setup (Partridge, 1982; Ballerini et al., 2008) or observed in their own ecological environment (Parrish and Edelstein-Keshet, 1999). Those experiments present the inconvenience to be costly to reproduce. Furthermore, the colossal lapse of evolutionary time needed to evolve swarming, make it almost impossible to study the emergence of such behavior experimentally.

Computer modeling has recently provided researchers with new, easier ways to test hypotheses on collective behavior. As it is well known, simulating individuals on machines offers easy modification of setup conditions and parameters, tremendous data generation, full reproducibility of every experiment, and easier identification of the underlying dynamics of complex phenomena.

From Reynolds' boids to recent approaches

In a massively cited paper, Craig Reynolds (1987) introduces the *boids* model simulating 3D swarming of agents called *boids* controlled only by three simple rules:

• Alignment: move in the same direction as neighbours

¹Copyright Walter Baxter and licensed for reuse under this Creative Commons Licence.

- Cohesion: Remain close to neighbours
- Separation: Avoid collisions with neighbours

Various works have since then reproduced swarming behavior, often by the means of an explicitly coded set of rules. For instance, Mataric (1992) proposes a generalization of Reynolds' original model with an optimally weighted combination of six basic interaction primitives². Hartman & Benes (2006) come up with yet another variant of the original model, by adding a complementary force to the *alignment* rule, that they call *change of leadership*. Unfortunately, in spite of the insight this kind of approach brings into the dynamics of swarming, it shows little about the pressures leading to its emergence. Many other approaches are based on informed agents or fixed leaders (Cucker & Huepe 2008, Su et al. 2009, Yu et al. 2010).

For that reason, experimenters attempted to simulate swarming without a fixed set of rules, rather by incorporating into each agent an artificial neural network brain that controls its movements. The swarming behavior is evolved by copy with mutations of the chromosomes encoding the neural network parameters. By comparing the impact of different selective pressures, this type of methodology eventually allows to analyze the evolutionary emergence of swarming.

Tu and Terzopoulos (1994) have swarming emerge from the application of artificial pressures consisting of hunger, libido and fear. Other experimenters have studied prey/predator systems to show the importance of sensory system and predator confusion in the evolution of swarming in preys (Ward et al. 2001, Olson et al. 2013).

In spite of many pressures hypothesized to produce swarming behavior, designed setups presented in the literature are often complex and specific. Previous works typically introduce models with very specific environments, where agents are given specialized sensors sensitive. While they are bringing valuable results to the community, one may wonder about systems with a simpler design.

In addition, even when studies focus on fish or insects that swarm in 3D (Ward et al. 2001) most keep their model in 2D. While the swarming can be considered to be similar in most cases, the mapping from 2D to 3D is found to be nontrivial (Sayama 2012). Indeed, the addition of a third degree of freedom may enable agents to produce significantly distinct and more complex behaviors.

Signaling agents in a resource finding task

This paper studies the emergence of swarming in a population of agents using a basic signaling system, while performing a simple resource gathering task. Simulated agents move around in a three dimensional space, looking for a vital but invisible food resource randomly distributed in the environment. The agents are emitting signals that can be perceived by other individuals' sensors within a certain radius. Both agent's motion and signaling are controlled by an artificial neural network embedded in each agent, evolved over time by an asynchronous genetic algorithm. Agents that consume enough food are enabled to reproduce, whereas those whose energy drops to zero are removed from the simulation.

During the first phase, we observe that agents progressively coordinate into clustered formations, which are preserved through the second phase. Such patterns do not appear in control experiments having the simulation start directly from the second phase, with the absence of resource locations. If at any point the signaling is switched off, the agents immediately stop swarming together. They start swarming again as soon as the communication is turned back on. Furthermore, it is observed that simulations with signaling lead to agents gathering very closely around food patches, whereas the control simulations with silented agents end up with them wandering around erratically.

The main contribution of this work is to demonstrate that collective motion can originate, without explicit central coordination, from the combination of a generic communication system and a simple resource gathering task. A specific genetic algorithm with an asynchronous reproduction scheme is developed and used to evolve the agents' neural controllers. In addition, the search for resource is shown to improve from the agents clustering, eventually leading to the agents gathering closely around goal areas. An indepth analysis shows increasing information transfer between agents throughout the learning phase, and the development of leader/follower relations that eventually push the agents to organize into clustered formations.

The rest of the paper is organized as follows. The next section describes the details of our model. Then simulation settings and results are discussed, before finally drawing a conclusion in the last section.

Model

Agents in a 3D world

We simulate a group of agents moving around in a cubic, toroidal arena of $600 \times 600 \times 600$. The agents rely on energy to survive. If at any point an agent's energy drops to zero, it is immediately removed from the environment. The task for the agents is to get as close as possible to a preset resource spot. By getting close to one of those spots, agents can gain more energy, allowing them to counterbalance the energy losses due to movement and signaling. An agent whose energy drops to zero is removed from the simulation. In this regard, the energy also represents each agent's fitness, and in this paper both terms are used interchangeably.

²Namely, those primitives are collision avoidance, following, dispersion, aggregation, homing and flocking.

The agent's position is determined by three floating point coordinates between 0.0 and 600.0. Each agent is positioned randomly at the start of the simulation, and then moves at a fixed speed of 1 unit per iteration. The direction of motion is decided by two motors controlling Euler angles ψ for pitch (i.e. elevation) and θ for yaw (i.e. heading).

Communication among agents

Every agent is also provided with one communication actuator capable of sending signals with intensities (signals are encoded as floating point values ranging from 0.0 to 1.0), and six communication sensors allowing it to detect signals produced by other agents up to a distance of 100 from 6 directions, namely frontal (0,1,0), rear (0,-1,0), left (1,0,0), right (-1,0,0), top (0,0,1) and bottom (0,0,-1)). The communication sensors are implemented so that every source point in a 100-radius sphere around the agent is linked to one and only one of its sensors. The sensor whose direction is the closest to the signaling source receives one float value, equal to the sum of every signal emitted within range, divided by the distance, and normalized between 0.0 and 1.0.

A neural network inside each agent

The agent's neural controller is implemented by an Elman artificial neural network with 6 input neurons, encoding the activation states of the corresponding 6 sensors, fully connected through a 10-neuron hidden layer to 3 output neurons controlling the two motors and the communication signal emitted by the agent. The hidden layer is given a form of memory feedback from a 10-neuron context layer, containing the values of the hidden layer from the previous time step.

All nodes in the neural network take values between 0.0 and 1.0. All output values are also floating values between 0.0 and 1.0, the motor outputs are then converted to angles between $-\pi$ to π . The activation state of internal neurons is updated according to a sigmoid function.

An asynchronous reproduction scheme

Our model differs from the usual genetic algorithm paradigm, in that it designs variation and selection in an asynchronous way. The reproduction takes place continuously throughout the simulation, creating overlapping generations of agents. This allows for a more natural, continuous model, as no global clock is defined, that could bias or weaken the model.

Every new agent is born with an energy equal to 2.0. In the course of the simulation, each agent can gain or lose a variable amount of energy. At iteration t, the fitness function f_i for agent i is defined by $f_i(t) = \frac{r}{d_i(t)}$ where r is the reward value and d_i is the agent's distance to the

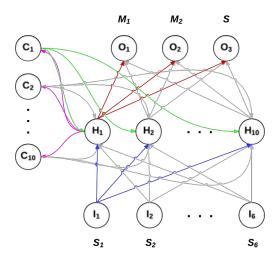


Figure 2: Architecture of the agent's controller, composed of 6 input neurons $(I_1 \ {\rm to} \ I_6)$, 10 hidden neurons $(H_1 \ {\rm to} \ H_{10})$, 10 context neurons $(C_1 \ {\rm to} \ C_10)$ and 3 output neurons $(O_1 \ {\rm to} \ O_3)$.

goal. The reward value is controlled by the simulation such that the population remains between 100 and 500 agents. All the way through the simulation, the agents also spend a fixed amount of energy for movement (0.01 per iteration) and a variable amount of energy for signaling costs $(0.001 \times signal\ intensity\ per\ iteration)$.

The weights of every connection in the neural network (apart from the links from hidden to context nodes, which have fixed weights) are encoded in a genotype and evolved through generations of agents. Each weight is represented by a unique floating point value in the genotype vector, such that the size of the vector is simply equal to the total number of connections in the neural network. The simulation uses a genetic algorithm with overlapping generations to evolve the weights of the neural networks. Whenever an agent accumulates 10.0 in energy, a replica of itself (with a 5% mutation in the genotype) is created and added to a random position in the arena. The agent's energy is decreased by 8.0 and the new replica's energy is set to 2.0. The choice for random initial positions is to avoid biasing the proximity of agents, so that reproduction does not become a way for agents to create local clusters.

Indeed, a local reproduction scheme (i.e. giving birth to offspring close to their parents), leads rapidly to an explosion in population size, as the agents that are close to the resource create many offspring that will be very fit too, thus able to replicate very fast as well. This is why the newborn offspring is placed randomly in the environment.

For our genetic algorithm to be effective, the number of agents must be maintained above a certain level. Also, the computation power limits the population size. The fitness allowed to the agents is therefore adjusted in order to maintain an acceptable number of agents (between 100 and 1000) alive throughout the simulation. In addition, agents above a certain age (5000 time steps) are removed from the simulation, to keep the evolution from standing still.

Results

Emergence of swarming

We observe agents coordinate together in clustered groups. As shown in Figure 3 (top) the simulation goes through three distinct phases. In the first one, agents wander in an apparently random way across the space. Then, during the second phase, agents progressively cluster into a rapidly changing shape, reminiscent of animal flocks³. In the third phase, towards the end of the simulation, the flocks get closer and closer to the goal⁴, forming a compact ball around it.

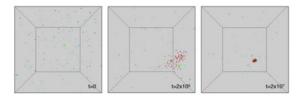


Figure 3: Visualization of the development of a typical run with a single resource spot. The agents start off in a random motion (left), then progressively come to coordinate in a dynamic cluster (middle), and finally flock more and more closely to the goal (right).

Figure 4 shows more in detail the swarming behavior taking place in the second phase. The agents coordinate in a dynamic, quickly changing shape, continuously extending and compressing, while each individual is executing fast paced rotations on itself. Note that this fast looping seems to be necessary to the emergence of swarming, as all trials with slower rotation settings never achieved this kind of dynamics. One regularly notices some agents reaching the border of a swarm cluster, leaving the group, and most of the time ending up coming back in the heart of the swarm.

In spite of the agents needing to pay a cost for signaling (cf. description of the model), the signal keeps an average value between 0.2 and 0.5 during the whole experiment (in the case with signalling activated).

It is also noted that a minimal rotation speed is necessary for the evolution of swarming. Indeed, it allows the agent to

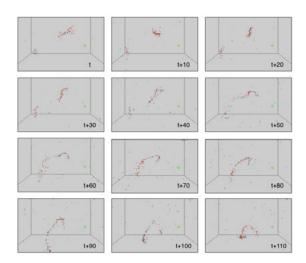


Figure 4: Visualization of the swarming behavior occurring in the second phase of the simulation.

react faster to the environment, as each turn making one sensor face a particular direction allows a reaction to the signals coming from that direction. The faster the rotation, the more the information gathered by the agent about its environment is balanced for every direction.

Neighborhood analysis

We choose to measure swarming behavior in agents by looking at the average number of neighbors within a radius of 100 distance around each agent. Figure 5 shows the evolution of the average number of neighbors, over 10 different runs, respectively with signaling turned on and off. A much higher value is reached around time step 10^5 in the signaling case, while the value remains for the silent control.

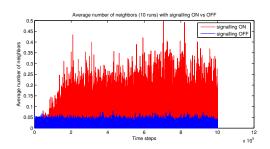


Figure 5: Comparison of the average number of neighbors (average over 10 runs, with 10^6 iterations) in the case signaling is turned on versus off

We also want to measure the influence of each agent on its neighborhood. To do so, the *inward* average transfer entropy

³As mentioned in the introduction, swarming can take multiple forms depending on the situation and/or the species. In this case, the clustering resemble in some aspects mosquito or starling flocking

⁴Even though results with one goal are presented in the paper, same behaviors are obtained in the case of two or more resource spots.

on agent's velocities is calculated⁵ between each neighbor within a distance of 100 and the agent itself. We will refer to this measure as inward neighborhood transfer entropy (NTE). This can be considered a measure of how much the agents are "following" their neighborhood at a given time step. The values rapidly take off on the regular simulation (with signaling switched on), while they remain low for the silent control, as we can see for example on Figure 6.

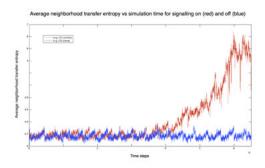


Figure 6: Plot of the average *inward* neighborhood transfer entropy for signaling on (*red curve*) and off (*blue curve*)

Similarly, we can calculate the *outward* neighborhood transfer entropy (i.e. the average transfer entropy from an agent to its neighbors). We may look at the evolution of this value through the simulation, in an attempt to capture the apparition of local leaders in the swarm clusters. Even though the notion of leadership is hard to define, the study of the flow of information is essential in the study of swarms. The single individuals' *outward* NTE shows a succession of bursts coming everytime from different agents, as illustrated on Figure 7. This frequent switching of the origin of information flow can be interpreted as a continual change of leadership in the swarm. The agents tend to follow a small number of agents, but this subset of leaders is not fixed over time.

On the upper graph in Figure 8, between iteration 10^5 and 2×10^5 , we see the average distance to the goal drop to values oscillating between roughly 50 and 300, that is the best agents reach 50 units away from the goal, while other agents remain about 300 units away. On the control experiment graph (Figure 8, bottom), we observe that the distance to the goal remains around 400.

Swarming, allowed by the signaling behavior, allows agents to stick close to each other. That ability allows for a winning strategy in the case when some agents already are successful at remaining close to a resource area. Swarming may also help agents find goals in the fact that they constitute an efficient searching pattern. Whilst an agent alone is subject to basic dynamics making it drift away, a bunch of

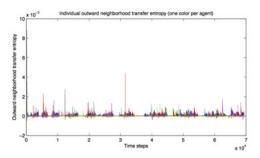


Figure 7: Plot of the individual *outward* neighborhood transfer entropy. Each color corresponds to a distinct agent.

agents is more able to stick to a goal area once it finds it, since natural selection will increase the density of surviving agents around a those areas. In the control experiments without signaling, it is observed that the agents, unable to form swarms, do not manage to gather around the goal in the same way as when the signaling is active.

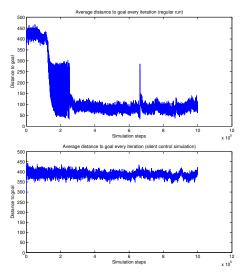


Figure 8: Average distance of agents to the goal with signaling (top) and a control run with signaling switched off (bottom)

Controller response

After simulation, we test neural networks of each swarming agent and qualitatively compare them to non-swarming ones'. We observed that characteristic shapes for the curve obtained with swarming agents presented a similarity (see Figure 9, top), and differed from the patterns of non-swarming agents (see Figure 9, bottom) which were also more diverse. In swarming individuals' neural networks,

⁵The calculations are analogous to Wibral et al. (2013).

patterns were observed leading to higher motor output responses in the case of higher signal inputs. This is characteristic almost every swarming individuals, whereas non-swarming agents present a wide range of response functions. A higher motor response may allow the agent to slow down its course accross the map by executing quick rotations around itself, therefore keeping its position nearly unchanged. If this behavior is adopted in the case where the signal is high, that is in presence of signaling agents, the agent is able to remain close to them.

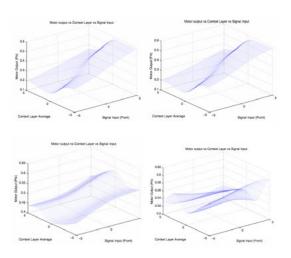


Figure 9: Plots of evolved agents' motor responses to a range of value in input and context neurons. The three axes represent signal input average values (right horizontal axis), context unit average level (left horizontal axis), and average motor responses (vertical axis). The *top* two graphs correspond to the neural controllers of swarming agents, and the *bottom* ones correspond to non-swarming ones'.

Phylogeny

To study the heterogeneity of the population, the phylogenetic tree is visualized (Figure 10). At the center of the graph is the root of the tree, which corresponds to time zero of the simulation, from which start the 200 initial branches. As those branches progress outward, they create ramifications that represent the descendance of each agent. The time step scale is preserved, and the segment drawn below serves as a reference for 10^5 iterations. Every fork corresponds to a newborn agent⁶. Therefore, every "fork burst" corresponds to a period of high fitness for the concerned agents.

On Figure 11, one can observe another phylogenetic tree, represented horizontally in order to compare it to the average

number of neighbors throughout the simulation. The neighborhood becomes denser around iteration 400k, showing a higher portion of swarming agents. This leads to a firstly strong selection of the agents able to swarm together over the other individuals, a selection that is soon relaxed due to the signaling pattern being largely spread, resulting in a heterogeneous population, as we can see on the upper plot, with numerous branches towards the end of the simulation.

The phylogenetic tree shows some heterogeneity, and the average number of neighbors is a measure of swarming in the population. The swarming takes off around iteration 400k, where there seems to be a genetic drift, but the signaling helps agents form and keep swarming.

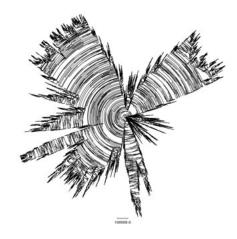


Figure 10: Phylogenetic tree of agents created during a run. The center corresponds to the start of the simulation. Each branch represents an agent, and every fork corresponds to a reproduction process.

Discussion

In our simulation, agents progressively evolve the ability to flock through communication to perform a foraging task. We observe a dynamical swarming behavior, including coupling/decoupling phases between agents, allowed by the only interaction at their disposal, that is signaling. Eventually, agents come to react to their neighbors' signals, which is the only information they can use to improve their foraging. This can lead them to either head towards or move away from each other. While moving away from each other has no special effect, moving towards each other, on the contrary, leads to swarming. Flocking with each other may lead agents to slow down their pace, which for some of them may keep them closer to a food resource. This creates a beneficial feedback loop, since the fitness brought to the agents will allow them to reproduce faster, and eventually multiply this type of behavior within the total population.

⁶The parent forks counterclockwise, and the newborn forks clockwise.

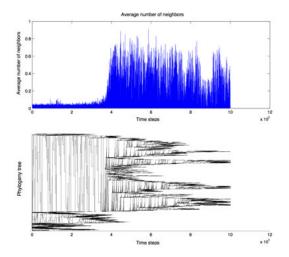


Figure 11: *Top:* average number of neighbors during a single run. *Bottom:* agents phylogeny for the same run. The roots are on the left, and each bifurcation represents a newborn agent.

In this scenario, agents do not need extremely complex learning to swarm and eventually get more easily to the resource, but rather rely on dynamics emerging from their communication system to create inertia and remain close to goal areas.

It should be noted that the simulated population has strong heterogeneity due to the asynchronous reproduction schema, which can be visualized in the phylogenetic tree (Figure 10). Such heterogeneity may suppress swarming but the evolved signaling helps the population to form and keep swarming. The simulations do not exhibit strong selection pressures to adopt specific behavior apart from the use of the signaling. Without high homogeneity in the population, the signaling alone allows for interaction dynamics sufficient to form swarms, which proves in turn to be beneficial to get extra fitness as it has been mentioned above.

The results presented in this paper can be compared to many works in the literature. Ward et al. (2001) and Olson et al. (2013) also show the emergence of swarming without explicit fitness, though those are based on a predator-prey model. The type of swarming obtained with simple pressures is usually similar to the one obtained in this study, that presents the advantage of being based on a very simple system based on resource finding and signaling/sensing. Models such as Blondel et al. (2005), Cucker & Huepe (2008) and Su et al. (2009) achieve swarming behavior based on explicit exchange of information from leaders. Our simulation improves on this kind of research in the sense that agents naturally switch leadership and followership by exchanging information over a very limited channel of communication.

Finally, our results also show the advantage of swarming for resource finding (it's only only through swarming, enabled by signaling behavior, that agents are able to reach and remain around the goal areas), comparable to the advantages of particle swarm optimizations (Kennedy et al. 1995), here emerging in a model with a simplistic set of conditions.

Conclusion

In this work we have shown that swarming behavior can emerge from a communication system in a resource gathering task. We implemented a three-dimensional agent-based model with an asynchronous evolution through mutation and selection. The results show that from decentralized leader-follower interactions, a population of agents can evolve collective motion, in turn improving its fitness by reaching invisible target areas.

Our results represent an improvement on models using hard-coded rules to simulate swarming behavior, as they are evolved from very simple conditions. Our model also does not rely on any explicit information from leaders, like previously used in part of the literature (Cucker & Huepe 2008, Su et al. 2009). It does not impose any explicit leader-follower relationship beforehand, letting simply the leader-follower dynamics emerge and self-organize. In spite of being theoretical, the swarming model presented in this paper offers a simple, general approach to the emergence of swarming behavior once approached via the boids rules.

References

- Ballerini, M., Cabibbo, N., Candelier, R., Cavagna, A., Cisbani, E., Giardina, I., Lecomte, V., Orlandi, A., Parisi, G., Procaccini, A., Viale, M., and Zdravkovic, V. (2008). Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study. Proceedings of the National Academy of Sciences, 105(4):1232–1237.
- Blondel, V., Hendrickx, J. M., Olshevsky, A., and Tsitsiklis, J. (2005). Convergence in multiagent coordination, consensus, and flocking. In *IEEE Conference on Decision and Control*, volume 44, page 2996. IEEE; 1998.
- Budrene, E. O., Berg, H. C., et al. (1991). Complex patterns formed by motile cells of escherichia coli. *Nature*, 349(6310):630– 633.
- Couzin, I. D. (2009). Collective cognition in animal groups. Trends in cognitive sciences, 13(1):36–43.
- Cucker, F. and Huepe, C. (2008). Flocking with informed agents. *Mathematics in Action*, 1(1):1–25.
- Czirók, A., Barabási, A.-L., and Vicsek, T. (1997). Collective motion of self-propelled particles: Kinetic phase transition in one dimension. *arXiv* preprint cond-mat/9712154.
- Huth, A. and Wissel, C. (1992). The simulation of the movement of fish schools. *Journal of theoretical biology*, 156(3):365–385.
- Kennedy, J., Eberhart, R., et al. (1995). Particle swarm optimization. In *Proceedings of IEEE international conference on*

- neural networks, volume 4, pages 1942-1948. Perth, Australia.
- Mataric, M. J. (1992). Integration of representation into goaldriven behavior-based robots. *Robotics and Automation*, *IEEE Transactions on*, 8(3):304–312.
- Olson, R. S., Hintze, A., Dyer, F. C., Knoester, D. B., and Adami, C. (2013). Predator confusion is sufficient to evolve swarming behaviour. *Journal of The Royal Society Interface*, 10(85):20130305.
- Parrish, J. K. and Edelstein-Keshet, L. (1999). Complexity, pattern, and evolutionary trade-offs in animal aggregation. *Science*, 284(5411):99–101.
- Partridge, B. L. (1982). The structure and function of fish schools. *Scientific american*, 246(6):114–123.
- Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In ACM SIGGRAPH Computer Graphics, volume 21, pages 25–34. ACM.
- Sayama, H. (2012). Morphologies of self-organizing swarms in 3d swarm chemistry. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, pages 577–584. ACM.
- Shimoyama, N., Sugawara, K., Mizuguchi, T., Hayakawa, Y., and Sano, M. (1996). Collective motion in a system of motile elements. *Physical Review Letters*, 76(20):3870.
- Su, H., Wang, X., and Lin, Z. (2009). Flocking of multi-agents with a virtual leader. *Automatic Control, IEEE Transactions* on, 54(2):293–307.
- Tu, X. and Terzopoulos, D. (1994). Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 43–50. ACM.
- Ward, C. R., Gobet, F., and Kendall, G. (2001). Evolving collective behavior in an artificial ecology. Artificial life, 7(2):191–209.
- Wibral, M., Pampu, N., Priesemann, V., Siebenhühner, F., Seiwert, H., Lindner, M., Lizier, J. T., and Vicente, R. (2013). Measuring information-transfer delays. *PloS one*, 8(2):e55809.
- Yu, W., Chen, G., and Cao, M. (2010). Distributed leader-follower flocking control for multi-agent dynamical systems with time-varying velocities. Systems & Control Letters, 59(9):543–552.
- Zaera, N., Cliff, D., and Janet, B. (1996). Not) evolving collective behaviours in synthetic fish. In *Proceedings of International* Conference on the Simulation of Adaptive Behavior. Citeseer.

Exploring conditions that select for the evolution of cooperative group foraging

Patrick B. Haley^{1,3}, Randal S. Olson^{2,3}, Fred C. Dyer^{2,3}, and Christoph Adami^{2,3}

¹The University of Texas at Austin, Austin, TX 78712
 ²Michigan State University, East Lansing, MI 48824
 ³BEACON Center for the Study of Evolution in Action, East Lansing, MI 48824
 patrick.haley@utexas.edu, olsonran@msu.edu, fcdyer@msu.edu, adami@msu.edu

Extended Abstract

Many prey choose to live, forage, and reproduce in groups — this is one of the most readily-observed phenomena in biology. Group living is potentially costly (because of competitive interactions among other reasons), and the benefits that outweigh these costs are difficult to understand, as they may interact in complicated ways (Krause and Ruxton, 2002). Collective vigilance is one oft-cited benefit of grouping behaviors. This claim relies on the principle that at each moment in time prey must make a choice between two mutually exclusive actions: foraging for food or being vigilant to look for predators. Group foraging potentially allows individuals to increase their foraging efficiency — and therefore their fitness — by sharing the expensive task of looking out for predators. Since isolating such decision-making in biological systems is difficult (particularly on an evolutionary timescale), we use digital organisms to study how this decision is made by groups of prey under the threat of predation.

Decision making in groups can take two different forms. Most computational research has concentrated on homogeneous groups, where every agent has an identical method of making decisions, usually regarding movement (Ward et al., 2001). However, it is also possible for groups to be heterogeneous, with behaviors varying drastically among individuals. A group with both altruistic and selfish members would be an example of such heterogeneity.

Here, we explore the effects of this distinction between homogeneous and heterogeneous group composition in relation to the evolution of vigilance behaviors. In addition, we examine the impact of two reproductive strategies: populations are either iteroparous (reproducing repeatedly) or semelparous (reproducing a single time before they die), which should affect the intensity of selection on antipredatory traits that influence survival to reproductive maturity.

Methods As in previous work, agent fitness is determined in a disembodied simulation, where the agent's goal is to forage as much as possible while surviving predator attacks (Ruxton and Beauchamp, 2008). All agents are in the same group, and group size is varied between experiments to explore its effect. Fitness is equal to the number of updates spent foraging rather than being vigilant. When the simu-

lation is complete, a genetic algorithm generates the next population of genotypes. An agent's genotype codes for a Markov Network in which one output state is the decision to forage or be vigilant (Olson et al., 2013). Currently, this means vigilance levels are defined by probabilities, but future studies could consider more complex outputs (e.g. multiple interrelated strategies), or inputs from a visual system that provides a richer picture of the environment.

In the simulation, vigilance comes into play when a predator attacks. The rate of this attack is relative to the size of the group, so there is no dilution of the attack risk with increasing group size. When the predator appears, it randomly selects one of the prey as its target. The predator then waits for several turns (approximating the time required to close for an attack). If the target is vigilant at some point in this interval, it becomes aware of the predator and has a 90% chance of survival. If the target remains unaware but another prey is vigilant, the target has a 50% chance of survival (i.e., the threat is automatically communicated). If none of the prey are vigilant, the target has a 10% survival rate.

The fitness function can be varied in two ways. First, prey can be evaluated in either homogeneous or heterogeneous groups. In a homogeneous evaluation, one genotype is considered at a time, copies of its agent are made to fill the simulation, and the final fitness of all of these clones is averaged at the end. In a heterogeneous evaluation, genotypes compete against one another and the fitness an agent has at the end of the simulation is its fitness for that generation. Since differing genotypes can evolve to exploit the vigilance of others, thus lowering their own vigilance, we hypothesize that vigilance will be greater in homogeneous populations.

The second method for varying fitness functions is reproductive method. In semelparous treatments, a prey that died during the simulation is assigned a fitness of zero for that entire simulation (i.e., only surviving agents are said to reach reproductive maturity). In iteroparous treatments, dead prey cease to forage but they do not lose the fitness they have acquired during their lifetime (i.e., reproduction is continually occurring in the group). We hypothesize that semelparous treatments select more strongly for survival, therefore they

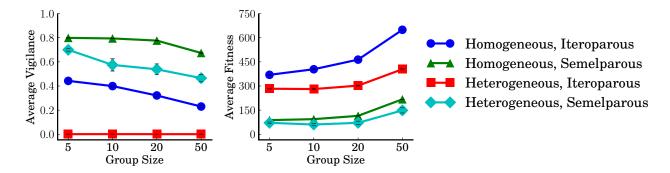


Figure 1: Semelparity and homogeneity select positively for vigilant behavior in prey without interfering with one another. Either treatment is sufficient to select for vigilance, but at least one is necessary. Fitness increases with group size while vigilance decreases, as predicted by the many-eyes hypothesis. Error bars are shown for 95% confidence intervals.

should produce higher levels of vigilance.

Results As shown in Figure 1, agents are more vigilant in semelparous treatments. This finding holds across both homogeneous and heterogeneous populations. In addition, homogeneous populations consistently evolve higher levels of vigilance than heterogeneous populations. We do not take this to mean that cooperative vigilance is impossible in heterogeneous groups, only that vigilance is a less stable strategy when genotypes compete against one another. This conclusion is supported by the evolution of vigilance in heterogeneous, semelparous populations, which shows that vigilance is possible outside of homogeneous conditions.

While homogeneity and reproductive strategy both select positively for vigilance, Figure 1 demonstrates that homogeneity plays the greater role. This can be seen from the lower evolved vigilance levels in the homogeneous, iteroparous populations compared to the heterogeneous, semelparous populations. Homogeneity and reproductive strategy also do not inhibit one another's selective pressure, since the homogeneous, semelparous population evolves the greatest levels of vigilance.

It is possible that some prey could evolve to take advantage of the vigilance of others. Such cheaters would spend most of their time foraging. Naturally, this is only possible in the heterogeneous treatment. We would expect such genotypes, if they exist, to be characterized by significantly higher fitness and lower vigilance values. However, in all cases the most fit organism in the population had vigilance and fitness values closely matching that of the rest of the population. Thus, we posit that this experimental environment is not suitable for the evolution of a stable cheating strategy.

Our general finding that vigilance promotes survival in groups even when it entails a tradeoff with foraging aligns with observational evidence of animals (Lima, 1995), which show higher levels of vigilance in smaller groups. The inverse relationship between vigilance and group size and the

direct relationship between fitness and group size we observed fit the predictions made by one theory on the benefits of grouping, the many-eyes theory (Pulliam, 1973). Such patterns suggest that the advantages of vigilance are a major driver of group living, and may favor living in larger groups in spite of the costs of competition. So far we have not explored such variables as the dilution of predation risk by group size (we explicitly controlled this), the dilution of vigilance costs by relatedness, or the ability to vary levels of vigilance conditional upon an assessment of the risks, but our system provides a platform for exploring such variables.

Acknowledgements

This research has been supported in part by the National Science Foundation (NSF) BEACON Center under Cooperative Agreement DBI-0939454. We thank Art Covert and the Freshman Research Initiative at UT Austin for their support. We gratefully acknowledge the support of the Michigan State University High Performance Computing Center and the Institute for Cyber-Enabled Research (iCER).

References

Krause, J. and Ruxton, G. D. (2002). *Living in Groups*. Oxford University Press, Oxford.

Lima, S. L. (1995). Back to the basics of anti-predatory vigilance: the group-size effect. *Animal Behaviour*, 49:11–20.

Olson, R. S., Hintze, A., Dyer, F. C., Knoester, D. B., and Adami, C. (2013). Predator confusion is sufficient to evolve swarming behaviour. *Journal of The Royal Society Interface*, 10.

Pulliam, H. R. (1973). On the advantages of flocking. *Journal of Theoretical Biology*, 38:419–422.

Ruxton, G. D. and Beauchamp, G. (2008). The application of genetic algorithms in behavioural ecology, illustrated with a model of anti-predator vigilance. *Journal of Theoretical Biology*, 250:435–448.

Ward, C. R., Gobet, F., and Kendall, G. (2001). Evolving collective behavior in an artificial ecology. *Artificial Life*, 7:191–209.

Swarm Grammars GD: Interactive Exploration of Swarm Dynamics and Structural Development

Sebastian von Mammen, Sarah Edenhofer

Organic Computing, University of Augsburg, Bavaria, Germany {sebastian.von.mammen, sarah.edenhofer}@informatik.uni-augsburg.de

Abstract

We present an interactive simulation of Swarm Grammars (SGs). SGs are an extension of L-Systems, where symbols of the production system are considered agents, whereas the given production rules determine their differentiation or reproduction. Assigning boid properties to the SG agents yields spatial dynamics apt to building structures in space and to collaborate stigmergically. In the presented interactive simulation, we put an emphasis on accessible interactive visuals for shaping the initial configuration of the simulation, to program the agents' perceptual and productive behavioural abilities, to dynamically drive developmental stages and to fine-tune visual structural properties such as colouring and scaling of the utilised developmental building blocks. Our system has been successfully deployed to promote swarm dynamics and developmental processes as important aspects of Artificial Life in a playful way. We present results from deploying the simulation in the context of an event to promote STEM research among high-school girls.

Introduction

Confronted with the challenge of providing an engaging entrance point to Artificial Life research to young high-school students, we decided on implementing an interactive Swarm Grammar (SG) simulation (von Mammen and Jacob (2009)). SGs seemed to be an adequate choice, as they integrate aspects of complex system dynamics—bridging from the behaviour or simple individuals to the emergent properties of large populations—and developmental processes—yielding unique artefacts that allow to trace spatial interaction processes over time.

In order to kindle intrinsically motivated engagement (Koster (2013)), the simulation was devised to (a) establish a relationship between the students, the software and its artefacts. (b) We had to provide accessible means to defining rules and configuring agents so the students could challenge their competence in the actual simulation processes and explore the outcome of their high-level programming ingenuity. (c) The intensity of the students' explorations and design efforts has to result from their voluntary engagement. Accordingly, the simulation provides an open-ended,

directly manipulatable playground environment that is freely and widely accessible as a public, WebGL-driven website¹.

In the remainder of this paper, we present the following aspects of the project. In the next section, we briefly summarise the key concepts from related works that we utilised in the agents' flocking definition and their (re-)production rules. Afterwards, we explain the simulation concept with an emphasis on its visual programming assets. Before concluding this work with a brief summary and an outlook on possible future work, we dedicate one section to presenting select simulation artefacts as well as preliminary user feedback.

Related Work

Our simulation concept has been developed to support the definition of perception and actuation properties of 'boid' agents and the construction, reproduction or differentiation of swarm grammar agents. The deployed mechanics of user interaction and visual programming techniques that will be outlined in the next section, are a translation of the following, underlying geometrical and grammatical relationships.

Boid Flocking

Reynolds (1987) presented the concept of 'bird-oids', or boids. It implements smooth, decentralised steering of swarms of agents based on several 'flocking urges' that emerge from the relationships between an agent and its neighbours. The neighbourhood is determined by the agent's radial field of view (FOV) defined with a maximal distance d_{max} and an angle α , see the annotated screenshot from our simulation in Figure 1.

Neighbours within a minimal distance d_{min} are considered 'too close' and trigger an evasive manoeuvre away from their centre (separation urge). The agent further synchronises its velocity (heading and speed) with the other neighbours (alignment urge) and it accelerates towards their centre (cohesion urge). Coefficients of these acceleration urges determine the emergent flocking behaviour. In addition to

¹http://www.vonmammen.org/SG-GD/

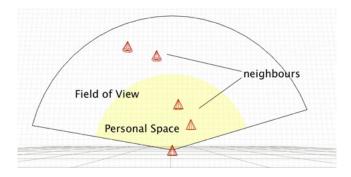


Figure 1: Perception of a 'boid' agent. Agents within its field of view are perceived as neighbours, those within the 'Personal Space' are considered too close, triggering an evasive manoeuvre.

separation, alignment and cohesion, we utilised a correspondingly weighted random vector (random urge) and a global direction vector (direction urge) to steer the individual agent. The acceleration \vec{a}_i of an individual i totals its j neighbour-dependent urges \vec{u}_{ij} , scaled by the individual's weights w_{ij} . The agents' acceleration and flight are kept within reasonable boundaries by maximal values for acceleration, a_{max} , and velocity, v_{max} . For simplicity sake, we chose integration step size $\Delta t = 1$ to infer the updated position p_i' of individual i. The corresponding equation system is listed below; markings denote the sequence of variable updates, $|\vec{x}|$ denotes the norm and \hat{x} the versor of vector \vec{x} .

$$\begin{array}{rcl} \vec{a}_{i} & = & \displaystyle \sum_{j} w_{ij} \vec{u}_{ij} \\ \\ \vec{a}'_{i} & = & \max(a_{max}, |\vec{a}_{i}|) \hat{\vec{a}}_{i} \\ \\ \vec{v}'_{i} & = & \vec{v}_{i} + \vec{a}_{i} \Delta t \\ \\ \vec{v}''_{i} & = & \max(v_{max}, |\vec{v}'_{i}|) \hat{\vec{v}}'_{i} \\ \\ \vec{p}'_{i} & = & p_{i} + \vec{v}''_{i} \Delta t \end{array}$$

In Figure 2, all flocking parameters are shown that the user is encouraged to alter. Similarly to Reynolds' later extensions (Reynolds (2000)), we give the user the ability to introduce novel agents, to change their FOV and flocking weights on the fly and, thus, to interactively guide the emerging flocks.

Swarm Grammar (Re-)Production

Originally, a swarm grammar $SG = (SL, \Delta)$ was conceived as a combination of a rewrite system $SL = (\alpha, P)$ and a set of agent specifications $\Delta = \{\Delta_{a_1}, \Delta_{a_2}, ... \Delta_{a_n}\}$ for n types of agents a_i (von Mammen and Jacob (2009)). The rewrite system SL loosely followed the concept of an L-system with axiom α and production rules P, as described by Prusinkiewicz and Lindenmayer (1990). In the simplest



Figure 2: Boid paramters as displayed to the user and offered for alteration. Changes to the Field of View parameters are immediately reflected by the neighbourhood visualisation of the introspected agent.

form of context-free 0L-systems, each rule has the form $p \to s$, where $p \in \Omega$ is a single symbol over an alphabet Ω , and $s \in \Omega^*$ is a word over Ω . The application of the replacement rule can be conditional, for instance upon a successful stochastic experiment (with specified probability θ) or repeatedly over time (with a specified time period Δt).

Agent specifications may include the flocking parameters described above such as the agents' FOVs and urge weights, as well as characteristic parameters of the geometrical objects they leave behind during their flight. Figure 3(a) shows a representative artefact produced by an early swarm grammar, emphasising the branching structure emerging from the reproduction rules $SL = \{A, \{A \rightarrow BBB, B \rightarrow A\}\}$ in combination with a moderate separation urge.

Later, the rule representation of Swarm Grammars was extended towards quantitative stigmergy (von Mammen and Jacob (2008a)), which (a) allowed to trigger (re-)production, in case a specific environment is perceived and (b) to utilise various static building blocks as well as agent progeny as product. Hence, spatial structures became the outcome of the agents' behavioural interactions rather than simply tracking their flight. A structure built by an extended SG is shown in Figure 3(b).

Interactive Simulation Concept

Our application offers several user interaction mechanisms that support the population and configuration of the simulation space. In order to keep the user interface free from clutter, we decided to omit certain functionalities altogether,





Figure 3: (a) An early SG definition emphasising branching (von Mammen and Jacob (2009)). (b) Artefact built by an extended, stigmergic SG (von Mammen et al. (2009)).

such as the rotation of geometric bodies, and we stripped the UI of any data that would not immediate benefit the target audience, such as the bodies' coordinates. At the backend, too, we pursued a simple but still ambitious management of agent specifications and geometric templates.

Basic Scene Manipulation

Figure 4 shows a close-up of the top-left corner of the main screen. Here, the user can start and stop the simulation. In the pull-down 'templates' menu an agent specification or a static geometry can be chosen to populate the simulation space. Clicking on any object in the scene (initially, there is the ground) places the selected template on top—in this way, the user can stack objects and populate in all three dimensions (Figure 5). Click and drag of an object moves it parallel to the ground. Hovering above an object and pressing the minus key removes an object (we found that the delete or backspace key is frequently assigned to other tasks in standard internet browsers).



Figure 4: The menu of the simulation's main screen. The simulation process can be started, feedback about the current simulation step is provided and a template can be selected to populate the simulation scene.

Right-clicking an object exposes its properties, as seen in Figure 5, and allows the user to change them. Property changes apply to all objects of the same name, which is shown as the top-most entry in the introspection menu to



Figure 5: Templates are placed on top of any clicked, existing objects. Properties of objects in the scene can be introspected and change on right click; an according menu appears in the upper-right corner of the screen.

the right. An alteration of a name triggers the creation of an according, new template in the 'templates' drop-down menu (Figure 4). In this way, the user can create a diverse set of static geometric objects and agent specifications.

Boid flocking parameters, including the field of view, and the (re-)production rules are part of an agent specification. As we offer a visual programming interface to configure some of these properties, the camera positions itself at a predefined distance from the agent when introspected. The dolly animation closing in on an introspected agent is shown in Figure 6.

Rule Editor

During introspection of an agent specification, alterations of the field of view parameters, d_{min} , d_{max} , α , result in an updated visual representation. This immediate reflection helps the user relate the parameter values to actual geometric dimensions and to quickly grasp the variables' relationships. Yet, the visualisation of the field of view plays another, more important role.

Figure 7 shows the view of an introspected agent. Production rules can be specified directly in the vicinity of the agent. In particular, dice, timers, and and arrow-enclosed timers can be dropped, which represent the abstract rule conditions and associate probability θ , point in time t, and time interval Δt variables, respectively. Platonic solids and agents that are placed outside of the introspected agent's FOV are products of a behavioural rule, those inside are considered quantitative, stigmergic conditions. Any such stigmergic conditions are fulfilled, if the according number of objects is perceived by the agent in the neighbourhood or in the personal space, respectively. For production objects, the minute geometric offset from the introspected agent is taken into account. Their displacement as expressed visually in the rule determines their relative placement in the simulation, which resolves the issue of potentially conflicting product placements.

As a consequence of the semantics associated with differently located platonic solids and agents, their placement and movement are diligently tracked and registered. The red bar

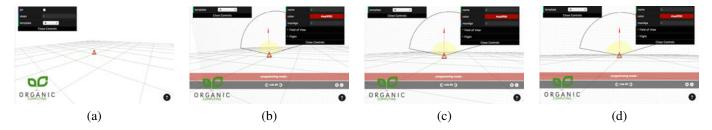


Figure 6: (a) A Swarm Grammar agent placed on the ground. When right-clicked, the camera automatically positions itself at a predefined distance (b-d).

towards the bottom of the screen provides feedback about the user's programming efforts and any corresponding ruleaffecting changes. The grey bar below provides the user with information about the currently displayed rule, to create new rules, to remove existing ones and to browse the complete set of rules of the introspected agent (and its namesakes).

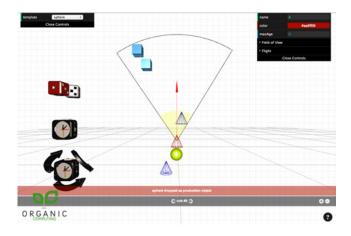


Figure 7: Introspecting an agent specification, sets of production rules may be visually programmed. Abstract conditions, stigmergic conditions, and products of the rules can be placed in the local vicinity of the agents.

Preliminary Feedback & Example Outcomes

In this section, we introduce the circumstances of the first deployment of the interactive simulation presented in this paper, *Swarm Grammars GD*. We provide details on the preliminary feedback we have gathered and we give examples of the artefacts built in this context.

Girls-in-STEM Programme

The concept was conceived while developing contributions to an entertaining and informative programme that aims at encouraging girls to develop and follow their passion for Science, Technology, Engineering and Mathematics (STEM) at the Faculty of Applied Computer Science at the University of Augsburg, Germany. As part of this programme, four groups of roughly ten girls between the ages twelve to fifteen attend four slots of 45 minutes each, offering different contents and activities: 'Autonomous Vehicles', 'Sight-Finder', 'Touch-Robots', and our entry 'Artificial Life'.

Aspects of Artificial Life Research

We identified the following aspects of Artificial Life research that our implementation makes accessible to interested novices in a playful manner.

Similarly to popular simulation environments such as NetLogo (Wilensky and CCL at Northwestern University (2014)), Swarm Grammars GD promotes an agent-based modelling approach. More specifically, it provides direct access, often supported by visual cues and interactive elements, to the parametric properties and sets of "if-then"rules that describe the behaviour of deterministically or stochastically acting, spatially interacting reactive agents (Wooldridge (2009)). When occurring in greater numbers, the interaction of such agents may result in complex feedback cycles, which in turn might lead to emergent phenomena, such as flocking dynamics (von Mammen and Jacob (2008b)) or complex built constructions (Bonabeau et al. (1999)). The means to directly program an individual agent or to simultaneously modify all agents of a certain type allows one to observe and experiment with the relationship between local behaviours and such global emergent patterns, as for instance portrayed by Johnson (2001).

As described in the section on Related Work, the interaction mechanisms that individual agents can perform in *Swarm Grammars GD* are limited to neighbourhood-dependent boid flocking (Reynolds (1987)) and rule-based production as in advanced swarm grammar concepts (von Mammen and Jacob (2009)). Both can be considered concrete concepts of two important Artificial Life themes, namely *collective locomotion* and *developmental models*. We make the first theme accessible by offering the means to visually program an agent's perceptional abilities. It is further promoted as simple construction rules that merely place single three-dimensional objects behind the agents at each simulated step effectively trace the resulting flight dy-

namics, as for instance seen in Figure 8(a).

Regarding the latter theme, developmental models, Swarm Grammars GD touches on the aspects of production, reproduction and differentiation, whereas these processes are triggered by the agents' internal states, effected by timers and stochastic experiments, as well as external stimuli (see our explanations of the visual rule editor above). Motivating differentiation based on locally perceived stimuli, such as the presence of a specific construction template or of a peer of a specific type, enables modelling a mechanism similar to task assignment in social insect societies (Camazine et al. (2003)). Stimulus-dependent construction efforts, on the other hand, allow one to implement sigmergic lines of communication (Grassé (1959)), i.e. indirect communication through the environment. All elements in Swarm Grammars GD, whether they are agents or built objects, have a lifespan attribute which determines the respective element's timely removal from the simulation (the elements are not removed, if this attribute is set to the value 0). Utilising such a timed appearance in combination with stigmergic construction and boid-based cohesion, a simplistic model of Ant Colony Optimisation can be retraced Dorigo (2007).

Presentation Sequence

Despite the intricate modelling options Swarm Grammars GD provide the user with, in the context of a short introduction to young novices, we directed our introductory STEM session to Artificial Life to the foundations of agent-based programming and discussed the intuitively accessible basics of swarm dynamics, construction and reproduction. After brief examples of L-Systems (Prusinkiewicz and Lindenmayer (1990)) and boids (Reynolds (1987)) and their utilisation as special effect techniques by the movie industry (schematic slides and movie snippets), we introduced Swarm Grammars SG hands-on. Following the structure of the above section 'Interactive Simulation Concept', we first explained the main view of the simulation space and basic user interactions to populate it. Next, we briefly demonstrated the exploration potential merely arising from altering various boid parameters. Finally, we detailed the composition of production rules (in this order: producing static geometries, initialising other agent specifications, and adding conditions to the rules). At this point, each group had approximately 25 to 30 minutes at its disposal for exploring and design artificial artefacts and swarm dynamics, under guidance and with feedback if desired. Our offer to print out screenshots of the individually generated artefacts was in good demand, we handed out 24 of them.

Leeway for Improvement

Some weaknesses of the current simulation became obvious during the supervised sessions—especially with respect to choosing reasonable parameter values, including boid urge weights, and configuring abstract production rule conditions such as chance or time steps. One could mitigate the issue of conflicting or ineffective boid parameter sets by offering several presets such as the ones evolved by Kwong and Jacob (2003). Regarding conditional values, if one does not want to drastically limit the expressiveness of rule compositions, warnings could be issued that hint at potentially unreasonable parameters. For instance, agent multiplication at high frequencies quickly exhausts the host computer, if the maximal life span of the agents is relatively high.

While exploring, one student asked how one could program the cubes (as opposed to the agents) to reproduce themselves. This question made clear that the distinction between producing agents and static geometries is arbitrary, not necessary, and possibly not even beneficial for the sake of functional distinction. At least one could expand the computational representation and nullify this rigid distinction. Other, more frequently asked questions were related to increasing the diversity of templates: although creating new templates by entering new names was quickly understood, this mechanism seemed to be too lengthy for supporting the creative diversity in colour and scale some users would have liked to deploy.

Despite the shortcomings of the current implementation, twelve out of a total of 42 participants declared our session and the use of *Swarm Grammars GD* to be the highlight of the whole introductory programme.

Example Artefacts

Figure 8 shows an array of six different Swarm Grammar artefacts programmed by participants of our session. We can identify different classes of structural complexity based on the flocking and production rule complexities of the SG agents. With only minor changes to the default boid flocking parameters (see Figure 2) and continuously dropping geometries, swarm motion is captured by the built artefacts (Figures 8 (a-c)). Agents with distinct construction behaviours—either resulting from differentiated reproduction or from interactively adjusting agent specifications—yield more visually complex structures (Figures 8(d-f)).

Perspective Shots

Unfortunately, as the students did not have much time to explore, play and create, their artefacts were mostly captured from a global perspective, which usually does not emphasise their appealing peculiarities. Figure 9 displays several Swarm Grammar configurations, snapshots of the emergent generative processes and close-ups of the final products set in scene.

In particular, three different Swarm Grammars are shown. The first one, with captions subtitled *cloud*, works based on a simple, unconditional production rule that traces an upwards-flocking agent with a cluster of spheres. The twists and turns triggered by the interplay of several flocking

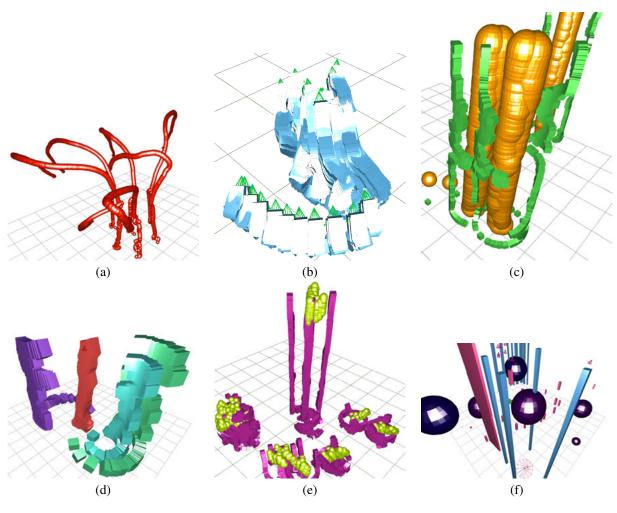


Figure 8: (a-b) Flocking SG agents leaving traces of one platonic solid (red spheres and blue cubes, respectively). (c) Individuals with a strong directional upwards urge leave a trail of two solids, golden spheres and offset green cubes. (d-f) More intricate and diverse structures emerge from building efforts by heterogeneous agent sets.

agents (as seen in Figure 9(b)) are exalted in the close-up by an upwards perspective and light shading. The second example deploys two agents, the first one simply flies upwards, repeatedly creating an offspring of a different kind ($\operatorname{rule}_{wall}^0$). The latter one is pushing hard to the right while continuously dropping cubic solids ($\operatorname{rule}_{wall}^1$) but it is also distracted by its neighbours and thrown off its path by some randomness. The resulting, aligned traces pave a solid uneven wall (close-up_wall); In the third example, a single agent is equipped with two rules, one to establish a continuous trace ($\operatorname{rule}_{tree}^0$) and the second one triggering periodic branching to two sides ($\operatorname{rule}_{wall}^1$). The repeated branching process (process_tree) yields a tree-like structure (close-up_tree).

Conclusion

In this paper, we presented an interactive Swarm Grammar simulation. It has been conceptualised and implemented in order to engage a young audience in Artificial Life concepts, namely swarm dynamics and developmental processes. The simulation attempts to intrinsically motivate the users by keeping the learning-curve as low as possible. At the same time, we challenge the users' competence by attaining a relatively expressive programmable representation, including boid flocking behaviour as well as (re-)production behaviour of Swarm Grammars. The gap between expressive representation and simplicity is bridged by means of visual programming interfaces for configuring the simulation space as well as individual agent behaviours.

We exhibited some of the artefacts designed by a number of high-school students at the age of twelve to fifteen. We suggested several possible improvements to the software based on feedback by the students but also based on observations during supervised hands-on sessions with the simulation. We complemented the display of the students' works by three additional Swarm Grammar examples that explicitly rely on (a) multiple construction elements in single rules, (b) differentiated reproduction, and (c) branching production rules.

In order to further the presented work, we suggest to translate all boid parameters into meaningful interactive visuals. For instance, the line-width of arrows representing various flocking urges could stand for the according, relative weights. Field of View and other visually represented parameters should be readily manipulatable, and not only be altered by means of textual GUIs. The maximal Age of an agent could be visualised by projecting a faded out geometry along its current trajectory. Also, the composition of (re-)production rules could be improved by relating them to the actual environment: the production objects could be projected on top of the actual simulation environment in order to facilitate precise definitions of environmental alterations.

Programmatically, we suggest switching from the current, class-based architecture to a component-based perspective that allows to aggregate behaviours. This would simplify the

template management and provide the flexibility to assign behaviours to arbitrary objects, as asked for by the students.

References

- Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). Swarm Intelligence: From Natural to Artificial Systems. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, New York.
- Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, J., Theraulaz, G., and Bonabeau, E. (2003). Self-Organization in Biological Systems. Princeton Studies in Complexity. Princeton University Press, Princeton.
- Dorigo, M. (2007). Ant colony optimization. *Scholarpedia*, 2(3):1461.
- Grassé, P.-P. (1959). La reconstruction du nid et les coordinations interindividuelles chezbellicositermes natalensis etcubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. *Insectes Soci*aux, 6(1):41–80.
- Johnson, S. (2001). *Emergence: The Connected Lives of Ants, Brains, Cities, and Software*. Scribner, New York.
- Koster, R. (2013). *Theory of fun for game design*. O'Reilly Media, Inc.
- Kwong, H. and Jacob, C. (2003). Evolutionary exploration of dynamic swarm behaviour. In *Congress on Evolutionary Computation*, Canberra, Australia. IEEE Press.
- Prusinkiewicz, P. and Lindenmayer, A. (1990). *The Algorithmic Beauty of Plants*. Springer, New York.
- Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34.
- Reynolds, C. W. (2000). Interaction with groups of autonomous characters. In *Game Developers Conference*, pages 449–460.
- von Mammen, S. and Jacob, C. (2008a). Evolutionary swarm design of architectural idea models. In *Genetic and Evolutionary Computation Conference (GECCO)* 2008, pages 143–150, Atlanta, USA. ACM Press.
- von Mammen, S. and Jacob, C. (2008b). The spatiality of swarms quantitative analysis of dynamic interaction networks. In *Proceedings of Artificial Life XI*, pages 662–669, Winchester, UK. MIT Press.
- von Mammen, S. and Jacob, C. (2009). The evolution of swarm grammars: Growing trees, crafting art and bottom-up design. *IEEE Computational Intelligence Magazine*, 4:10–19.
- von Mammen, S., Novakowski, S., Hushlak, G., and Jacob, C. (2009). Evolutionary swarm design: How can swarm-based systems help to generate and evaluate designs? *DESIGN Principles & Practices: An International Journal*, 3:371–386.
- Wilensky, U. and CCL at Northwestern University (2014). Netlogo: A cross-platform multi-agent programmable modeling environment. http://ccl.northwestern.edu/ netlogo/.
- Wooldridge, M. J. (2009). *An Introduction to MultiAgent Systems*. John Wiley and Sons Ltd, West Sussex, UK, 2nd edition.

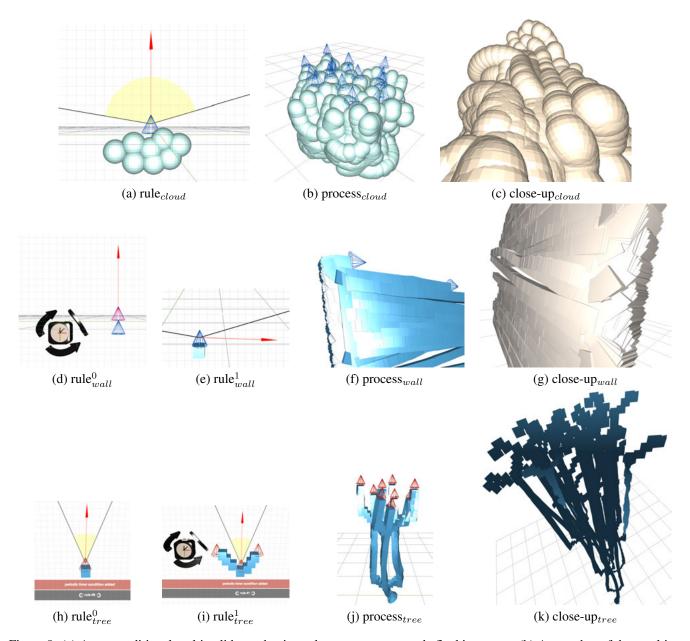


Figure 9: (a) An unconditional multi-solids production rule to trace an upwards-flocking agent. (b) A snapshot of the resulting developmental process, given a small set of initial agents. (c) A close-up of the final artefact. (d) An agent that flies upwards without considering any distractions and periodically produces (e), an agent heading to the right and leaving a cubic-trail behind. Its trajectory is influenced by its neighbours and chance. (f) The resulting developmental process, and (g) the final artefact close-up. (h) A simple trail production rule, combined with (i) a periodic branching rule, resulting in a (j) branching developmental process. (k) The final artefact set in scene with a pixelated 2D style.

Four Classes of Morphogenetic Collective Systems

Hiroki Sayama

Collective Dynamics of Complex Systems Research Group Binghamton University, State University of New York, Binghamton, NY 13902-6000, USA sayama@binghamton.edu

Abstract

We studied the roles of morphogenetic principles heterogeneity of components, dynamic differentiation/redifferentiation of components, and local information sharing among components—in the self-organization of morphogenetic collective systems. By incrementally introducing these principles to collectives, we defined four distinct classes of morphogenetic collective systems. Monte Carlo simulations were conducted using an extended version of the Swarm Chemistry model that was equipped with dynamic differentiation/re-differentiation and local information sharing capabilities. Self-organization of swarms was characterized by several kinetic and topological measurements, the latter of which were facilitated by a newly developed network-based method. Results of simulations revealed that, while heterogeneity of components had a strong impact on the structure and behavior of the swarms, dynamic differentiation/re-differentiation of components and local information sharing helped the swarms maintain spatially adjacent, coherent organization.

Introduction

Self-organizing behaviors of biological collectives have been subject to many scientific inquiries (Reynolds, 1987; Ben-Jacob, Cohen & Gutnick, 1998; Parrish & Edelstein-Keshet, 1999; Camazine et al., 2000; Sole & Goodman, 2000; Vicsek & Zafiris, 2012). They have also been enthusiastically applied to engineering problem solving as a new paradigm and methodology for decentralized, distributed problem solving by collaborative artificial agents (Kennedy & Eberhart, 1995; Bonabeau, Dorigo, & Theraulaz, 1999; Leonard & Fiorelli, 2001; Dorigo et al., 2001–2005; Pfeifer, Iida, & Bongard, 2005; Engelbrecht, 2005; Baldassarre, Parisi, & Nolfi, 2006; Braha, Minai, & Bar-Yam, 2006; Doursat, 2008, 2011). Typical assumptions made in the existing literature are that the system components interact with each other mostly locally and make decisions at individual levels, which eventually leads to the emergence of non-trivial (and potentially useful) macroscopic behaviors. Those models have been successful in reproducing various self-organizing patterns and adaptive functionalities.

However, theoretical models used in earlier studies were predominantly focused on homogeneous physical collectives and animal populations. While their simplicity is by itself a virtue in some regard (Vicsek & Zafiris, 2012), they are often too simple to capture more complex phenomena seen in real-world biological collectives, such as multi-cellular organisms' morphogenesis and physiology, termite colony building and maintenance, and growth and self-organization of human social systems. These systems operate with highly sophisticated within-system regulation mechanisms, or "programs" (Doursat, 2008, 2011). What are common among those real-world complex biological collectives are heterogeneity of components, dynamic differentiation/re-differentiation of components (i.e., dynamic switching of component types/roles), and local information sharing among components that influence their differentiation. These morphogenetic principles are often absent in earlier models of biological collective behaviors, and they have not been fully utilized in engineering applications either. Here we define a morphogenetic collective system as a system made of a large number of components that self-organize to form nontrivial structures and behaviors using these morphogenetic principles.

There is growing literature of analytical and numerical studies on self-organizing behavior of biological collectives (Vicsek & Zafiris, 2012; Vicsek et al., 1995; Mogilner & Edelstein-Keshet, 1998; Kunz & Hemelrijk, 2003; Hemelrijk & Kunz, 2005; D'Orsogna et al., 2006; Szabo et al., 2006; Chuang et al., 2007; Newman & Sayama, 2008; Paley et al., 2008; Chate et al., 2008; Romanczuk, Couzin, & Schimansky-Geier, 2009; Tian et al., 2009; Bernoff & Topaz, 2011). It is repeatedly reported that there are a small number of universal classes of collective behaviors, such as disordered, highly ordered, rotational, critical, and jamming patterns, as well as various forms of phase transitions between those classes (Vicsek & Zafiris, 2012). Most of these results are based on the assumption that collectives are homogeneous in terms of their components' kinetic and behavioral properties. Within-population variations are rarely considered in those theoretical models.

There are some studies that considered the effects of heterogeneity within biological collectives. Graves et al. studied mixed-species bird flocks in Amazonia and created a computational model of them (Graves & Gotelli, 1993). More recently, Couzin et al. studied self-sorting of a fish school caused by physical variations among individuals (Couzin et al., 2002). His group also studied collective decisions of a swarm influenced by a small number of informed individuals, or leaders (Couzin et al., 2005). We also proposed the Swarm Chemistry model (Sayama, 2009, 2010, 2012a,b) to explore self-organization of swarms made of kinetically distinct types of particles. However, these studies still assumed that within-population variations are variations of fixed individual properties. None of them considered a more sophisticated form of dynamic, adaptive changes of behavioral rules of individuals within a population, potentially through local communication and information sharing.

When one looks at real-world biological collectives, there are a number of examples where more complex forms of heterogeneous collectives produce highly intricate patterns and behaviors that look almost self-evidently "designed" by someone or something (Doursat, 2008, 2011; Turner, 2007). Such instances can be found at every scale in biology. For example, Ben-Jacob et al. reported very complex, heterogeneous, even intelligent, information processing and motion control taking place in bacteria societies (Ben-Jacob, Cohen & Gutnick, 1998; Ingham & Ben-Jacob, 2008; Ben-Jacob, 2009). Social insects are another well-studied example, where dynamic switching of different roles driven by local information sharing realizes highly efficient division of labor (Camazine et al., 2000; Bonabeau, Dorigo, & Theraulaz, 1999; Bonabeau et al., 1997; Campos et al., 2000; Beshers & Fewell, 2001). Among the most interesting and complex examples are the incredibly sophisticated, large-scale mounds built and maintained by termites (Turner, 2000, 2007, 2011). A termite mound operates as if it were a carefully designed and fully integrated physiological system, inside which individual termites "differentiate" to play various different roles. The complexity of such realworld biological collectives were not fully captured in the earlier literature of collective behaviors mentioned above.

Four Classes of Morphogenetic Collective Systems

As briefly reviewed above, the dynamics and capabilities of different types of morphogenetic collective systems are yet to be fully understood. Recent increase of studies that incorporate at least part of the morphogenetic principles indicates the promising nature of this direction of research. In order to systematically study the effects of each of the morphogenetic principles, here we propose the following four distinct classes of morphogenetic collective systems, which are obtained by incrementally introducing morphogenetic principles to agents' behavioral and communication capabilities

(Fig. 1):

- A. *Homogeneous collectives*, where agents' behaviors are determined by a globally defined, uniformly applicable function of observations.
- B. *Heterogeneous collectives*, where an agent's behavior is determined by a function of observations specified by the agent's static state or type.
- C. Heterogeneous collectives with dynamic differentiation/ re-differentiation, where the heterogeneity of agent behaviors is created and dynamically maintained by transitions of agents' internal states. Agents' state transitions are also determined by a function of observations and states.
- D. Heterogeneous collectives with dynamic differentiation/ re-differentiation and local information sharing, where agents can share information (internal states and their observations) with local neighbors, in addition to all the above capabilities.¹

Our rationale in defining these four classes is that each morphogenetic principle requires the precedent one. Namely, differentiation/re-differentiation requires multiple states or types of components (heterogeneity), and information sharing would make sense only if agents could change their behaviors according to it (differentiation/re-differentiation). We thus argue that these four classes should represent a natural, straightforward hierarchy of morphogenetic collective systems, arranged in the ascending order of their organizational complexity.

In this classification, Class A includes traditional collective behavior models, including Boids (Reynolds, 1987) and its variants that use homogeneous swarms. More recent models that use heterogeneous swarms (Couzin et al., 2002, 2005; Sayama, 2009, 2012a) belong to Class B. Examples that may be close to those of Class C are the variants of Swarm Chemistry (Sayama, 2010, 2011; Sayama & Wong, 2011) that implemented stochastic differentiation/redifferentiation of agents, though it was not driven by any state-transition function and thus not dynamical or adaptive. Finally, real biological/social morphogenetic systems—such as embryogenesis of multicellular organisms, colonies of eusocial insects and cities in human civilizations—are most likely in Class D. Several models proposed in Morphogenetic Engineering (Doursat, Sayama, & Michel, 2012) also belong here. However, we are not aware of any well established model of swarm-based collective behaviors that belong to this class.

¹Here we consider information sharing as explicit signal transmission among agents to share externally unobservable information about their internal states and observations with local neighbors. This should not be confused with kinetically transferred information theoretic waves propagating in swarms that were studied in recent literature (Wang et al., 2012).

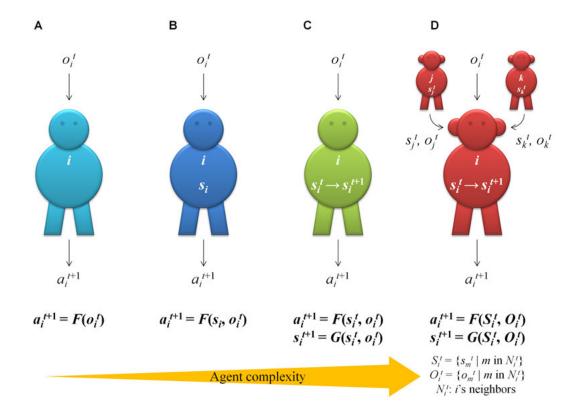


Figure 1: Four classes of morphogenetic collective systems proposed in this paper. Variables s_i , o_i and a_i represent the internal state of agent i, the observation it receives from the environment, and the corresponding action it takes (e.g., acceleration), respectively. A: Homogeneous collectives. Agents' behaviors are determined by a function of observations, $F(\circ)$. B: Heterogeneous collectives. Each agent has its own static state (s_i) , and F takes s_i as an additional argument as well as o_i . C: Heterogeneous collectives with dynamic differentiation. Agents' states can dynamically change according to another function, $G(\circ)$. D: Heterogeneous collectives with dynamic differentiation/ re-differentiation and local information sharing. Arguments of the functions are sets of s_i and o_i within the agent's neighborhood, which represents the local information sharing.

Model: Morphogenetic Swarm Chemistry

To study qualitative and quantitative differences in possible morphologies and behaviors between the four classes, we have developed a mathematical model of morphogenetic collective systems by implementing new rule-based state transition and local information sharing capabilities in the Swarm Chemistry model (Sayama, 2010, 2012b). Swarm Chemistry is naturally suitable for this research task because it already can represent both homogeneous and heterogeneous collective systems in its model framework.

In the extended model, the design of a swarm is specified in three parts: a recipe \mathcal{R} , a preference weight matrix U, and a local information sharing coefficient w. Definitions of these parts are as follows:

Recipe \mathcal{R} : A list of different kinetic parameter settings for multiple swarm states. Each entry in a recipe is composed of a relative frequency of a particular state within the

swarm and its kinetic parameter settings (e.g., local perception range, normal speed, strengths of kinetic forces, etc.). This part of the specification is the same as in our earlier studies (Sayama, 2009, 2010).

Preference weight matrix U: A $n \times (n+5)$ rectangular real-valued random matrix where n is the length of the recipe, i.e., the number of possible states of agents in a swarm. Its contents represent how each observation component (defined later) affects the agent's preference for a particular state choice.

Local information sharing coefficient w: A real number in [0, 1], which determines how much information about the neighbors' states and observations are to be shared and incorporated into the agent's own state transition.

Each agent in a swarm in this extended model has its own state s_i , in addition to position x_i and velocity v_i . The movement and state transition of agents is simulated as follows:

- **Step 1.** An agent computes its action (acceleration) based on its neighbors' relative positions and velocities using the standard Swarm Chemistry simulation algorithm (Sayama, 2009).
- **Step 2.** Before making any actual movement, the agent also computes an (n + 5)-dimensional *observation vector* o_i that summarizes the situation the agent is in. More details of this vector will be discussed later.
- **Step 3.** Once all the agents computed their actions and observation vectors, each agent updates its velocity according to the computed acceleration, and then moves using the updated velocity.
- **Step 4.** The agent computes a *state preference vector* $u_i = (1-w)Uo_i + wU\langle o \rangle_i$, where $\langle o \rangle_i$ is the average of observation vectors of other agents in the local neighborhood. If there are no other agents found in the neighborhood, $u_i = Uo_i$ regardless of w.
- **Step 5.** The agent checks if the s_i -th component of u_i , $u_i(s_i)$, is negative. If this is the case, it means that its current state is not preferable in the current situation, and therefore the agent attempts, with probability $1 \exp(u_i(s_i))$, to choose a new state. A new state will be chosen as the next value of s_i , via a roulette selection where $\exp(u(s))$ is used as the selection weight for state s_i .

The observation vector o_i plays an important role in this model. While there are many choices for how to construct an observation vector, we used the following definition as an initial step of our investigation. The first n components of o_i are all 0's except for the s_i -th component that is set to 1. This part embeds the information about the agent's current state into the vector so that it can be superposed and averaged with other agents' states. The remaining five components of o_i are as follows:

- $|\langle x \rangle_i x_i|^2/R_i^2$: Square of the relative distance from the average position of other agents in the neighborhood $(\langle x \rangle_i)$. R_i is the perception range in the parameter set the agent is currently using. If there are no other agents nearby, this is set to 0.
- v_i^2/v_{in}^2 : Square of the ratio of the agent's current velocity and the normal velocity (v_{in}) in the parameter set the agent is currently using.
- $\langle v \rangle^2 / v_{in}^2$: Square of the ratio of the neighbors' average velocity ($\langle v \rangle$) and the normal velocity in the parameter set the agent is currently using. If there are no other agents nearby, this is set to 0.
- $|\langle v \rangle_i v_i|^2/v_{in}^2$: Square of the relative difference in velocity between the agent and its neighbors. If there are no other agents nearby, this is set to 0.

• 1: A constant term.

Although it is a highly constrained, mathematically stylized formulation, this extended Swarm Chemistry model can still describe a wide variety of morphogenetic collective systems, including the four classes proposed in this paper. Specifically, including only one parameter set in $\mathcal R$ and letting U=0 and w=0 will make a Class A swarm. Including multiple parameter sets in $\mathcal R$ with U=0 and w=0 will make a Class B swarm. A Class C swarm will be obtained by additionally adopting a non-zero matrix for U while w=0. Finally, a Class D swarm will be obtained by adopting non-zero U and w.

Experiments

We conducted systematic Monte Carlo simulations to see if there were any significant differences in the dynamics of morphogenetic collective systems among the four classes. Specifications of swarm designs were configured as follows:

- The number of agents was fixed to 300 for all cases.
- The number of possible agent states n was set to 1 for Class A swarms, or otherwise $\xi + 2$ where ξ is a random integer sampled from a Poisson distribution with mean 1.
- Recipe \mathcal{R} was created by sampling each kinetic parameter's value from a uniform distribution between 0 and the parameter's maximum value defined in (Sayama, 2009). The relative frequencies of different parameter settings were determined by randomly dividing 300 into n portions.
- Preference weight matrix U was set to all 0's for Class A and B swarms. For Class C and D swarms, each component of U was sampled from a normal distribution with mean 0 and standard deviation 0.1.
- Local information sharing coefficient w was set to 0 for Class A, B and C swarms, while it was sampled from a uniform distribution between 0 and 1 for Class D swarms.

Five hundred independent simulation runs were conducted for each class. Each run was initialized with 300 agents whose positions were randomly distributed in a two-dimensional 300×300 (in arbitrary unit) square area and whose velocities were set to 0, and then simulated for 400 time steps. In each time step during the second half of the simulation (t=201-400), the following properties were measured from the agents' positions and velocities:

- Average speed of the swarm as a whole $(|\langle v \rangle|, \text{ where } \langle \ldots \rangle)$ denotes an average over all the agents hereafter unless noted otherwise).
- Average of absolute speed of agents $(\langle |v| \rangle)$.

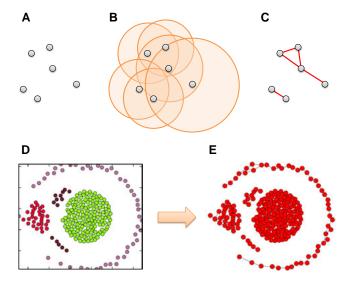


Figure 2: Construction of networks from swarms. Top: Illustration of the algorithm. A: Spatially distributed agents. B: Ranges of neighbor recognition drawn around the agents. The radius of each range is αd_i (see text). C: Resulting network where pairs of agents that mutually recognize each other as neighbors are connected. Bottom: An example of network construction from a simulated swarm. D: Original swarm snapshot. E: Network constructed from agent positions in D.

- Average angular velocity of the swarm as a whole $(\langle (x-\langle x\rangle) \times v/|x-\langle x\rangle|^2\rangle)$.
- Average distance of agents from center of mass $(\langle |x \langle x \rangle| \rangle)$.
- Average pairwise distance $(\langle |x_1 x_2| \rangle$, where x_1 and x_2 are positions of two randomly sampled agents; the average in this case was over 10,000 sampled pairs).

In addition, topological properties of the swarm morphology were also measured. For this task, we constructed a network for each swarm at each time step during the measurement period by connecting agents that were spatially close to each other. More specifically, we first measured a characteristic "neighbor" distance d_i for each agent by calculating the mean of its distances to its k nearest neighbors. The agent then recognized all other agents within distance αd_i as its "neighbors". Once this was done for all agents, the pairs of agents that mutually recognized each other as "neighbors" were actually connected. We used k=2 and $\alpha=1.7$, which were empirically chosen to approximate cluster structures recognized visually by human experimenters. Figure 2 shows examples of this network construction process.

For each swarm converted to a network, we measured the following topological properties:

• Number of connected components.

- Average size of connected components.
- Homogeneity of sizes of connected components. This was measured by the normalized entropy in the distribution of sizes of connected components. If there was only one connected component, this was set to 1.
- Size of the largest connected component.
- Average size of connected components smaller than the largest one. If there was only one connected component, this was set to 0.
- Average clustering coefficient.
- Link density.

Altogether, we obtained time series of 5+7=12 measurements from each simulation run. These time series were further summarized by calculating their respective mean and standard deviation over time. As a result, the structure and behavior of each swarm was characterized by $2\times 12=24$ outcome variables. Finally, a visual image of the swarm at the end of the simulation (t=400) was also recorded for visual inspection of the swarm's topology.

The simulator was implemented in Python using NetworkX (Hagberg, Schult, & Swart, 2008) and PyCX (Sayama, 2013). The code is available upon request.

Results

Simulation results are summarized in Table 1, where medians of 24 outcome variables were shown for each class together with *p*-values obtained using the Kruskal-Wallis median test between the four classes. Statistically significant differences were detected for most of the outcome variables, especially for topological outcome variables, which demonstrates the effectiveness of our network-based topology characterization method. Most of the variables with statistical significance showed clear differences between Class A (homogeneous) and other (heterogeneous) swarms. The temporal mean of the average clustering coefficient was the only outcome variable that did not show statistically significant differences between the classes. This metric may be primarily determined by constraints built in our simulation or network construction algorithms.

We found several notable patterns in Table 1. First, the medians of Class A swarms consistently took the lowest values among the four classes with regard to the temporal standard deviations of measurements (lower half of Table 1). This indicates that the interactions between different types of agents helped produce dynamic behaviors, causing temporal fluctuations of their macroscopic properties. Second, there were clear differences between Class B (with no state

²Visual snapshots of the swarms' final configurations for each of the four classes are also available online at http://bingweb.binghamton.edu/sayama/SwarmChemistry/.

Table 1: Comparison of medians of 24 outcome variables between four classes of morphogenetic collective systems. The p-values of the Kruskal-Wallis median test were shown in the rightmost column (*: p < 0.01, **: p < 0.001, ***: p < 0.0001). Variables with statistically significant differences were highlighted in color (yellow: high, cyan: low). The intensity of color is adjusted according to the p-value of the variable and the distance from overall average. The top half shows temporal means of the 12 measurements while the bottom half shows temporal standard deviations of the 12 measurements.

				Class A median	Class B median	Class C median	Class D median	K-W test p-value
		ic nes	average speed of swarm	2.46921	3.63557	3.71434	4.0491	0.00006***
			average absolute speed of agents	7.93488	9.14903	9.15137	9.86948	0.00024**
		Kinetic outcomes	average angular velocity of swarm	0.000381388	0.00135723	0.000879347	0.000775047	0.00000***
۰	u	out E	average distance from center	144.646	375.673	215.245	206.946	0.00000***
	[ea]		average pairwise distance	197.277	521.194	302.728	284.57	0.00000***
	Temporal Mean		number of connected components	1.54	11.65	9.91	7.8175	0.00000***
	ıra		average size of connected components	234.013	27.7284	32.4974	41.5561	0.00000***
	np(es	homogeneity of connected component sizes	0.957634	0.647418	0.669637	0.670815	0.00000***
	Ler	logi om	size of largest connected component	298.633	177.26	199.48	212.645	0.00000***
		Topological outcomes	average size of smaller connected components	1.00075	6.93472	5.18955	5.0891	0.00000***
			average clustering coefficient	0.43364	0.434347	0.433522	0.431574	0.18211
			link density	0.0164127	0.0131673	0.0137475	0.0136435	0.00000***
			average speed of swarm	0.109343	0.318225	0.28713	0.262987	0.00000***
	00	Kinetic outcomes	average absolute speed of agents	0.0285353	0.0849336	0.0910742	0.0853709	0.00000***
	iati	Kinetic utcome	average angular velocity of swarm	0.00185723	0.00357737	0.00376014	0.00401965	0.00000***
	evi	M Out	average distance from center	0.353314	45.9475	11.2173	4.87346	0.00000***
	d D		average pairwise distance	1.11209	62.249	15.4285	6.60913	0.00000***
	lar		number of connected components	0.156125	2.17945	1.86601	1.79757	0.00000***
	Temporal Standard Deviation	_	average size of connected components	1.24983	<mark>4.90952</mark>	4.20666	4.12473	0.00000***
		ical ies	homogeneity of connected component sizes	0.0176554	0.0424783	0.0458559	0.053207	0.00000***
		log	size of largest connected component	0.211601	<mark>5.37309</mark>	4.24246	<mark>4.94907</mark>	0.00000***
		Topological outcomes	average size of smaller connected components	0.297179	1.54578	1.03091	1.04183	0.00000***
	Te		average clustering coefficient	0.00936916	0.0174722	0.0165864	0.0162297	0.00000***
			link density	0.000306748	0.000324602	0.000351948	0.000350871	0.00000***

transitions) and Classes C & D (with state transitions) regarding temporal standard deviations of the average distance of agents from the center of mass and the average pairwise distance. This is likely due to the fact that, while Class B swarms tend to disperse into smaller clusters easily, agents of Class C & D swarms can stay together and maintain spatially adjacent, coherent organization more often, because adaptive state transitions help initially incompatible agents assimilate into kinetically compatible types.

Finally, we noticed a consistent trend in a number of the outcome variables that the properties of Class C & D swarms sat somewhere in between those in Class A and Class B. This could also be understood in that dynamic state transition, possibly driven by local information sharing, has enabled swarms to adaptively achieve coherence in their structures and behaviors. However, the results also indicate that

the swarms in Classes C & D did not simply turn into a more homogeneous state like Class A ones, because nearly all the topological outcome variables showed significant differences between Class A and Classes C & D. Therefore, this apparent trend found in morphogenetic collective systems in Classes C & D must be understood not as simple homogenization but as an emergent shift of higher-level system properties.

Conclusions

In this paper, we proposed a classification of morphogenetic collective systems based on the absence or presence of three morphogenetic principles: heterogeneity of components, dynamic differentiation/re-differentiation of components, and local information sharing among components. Monte Carlo simulations with an extended morphogenetic

Swarm Chemistry model demonstrated that, while heterogeneity of components had a strong impact by itself on the structure and behavior of the swarms, the other two morphogenetic principles, i.e., dynamic differentiation/redifferentiation of components and local information sharing, greatly contributed to the maintenance of spatially adjacent, coherent organization of swarms. Interestingly, many outcome measurements of Class C and D swarms fell somewhere in between those of Class A and Class B. This result indicates that the dynamic, adaptive state transition of components possibly driven by their mutual information sharing is playing an essential role in achieving structural and functional integration of biological and social collectives.

The present study has several fundamental limitations. First, the dynamics of swarms were explored only through a limited number of random parameter sampling and analyzed only using simple median comparisons. It was a reasonable first step of exploration when nothing was known about the model, but this approach would not be able to characterize behavioral diversity and richness of each class of systems or to discover non-trivial behaviors that would be statistically rare but unique and interesting. To fully explore and understand the limit of dynamical diversity of each class, more sophisticated evolutionary or other population-based search methods should be conducted. Now that we have 24 outcome metrics defined, we can try evolving morphogenetic collective systems toward a certain area in this metric space, to examine how closely the swarms of each class can achieve the target properties.

Second, the model we used in this study (morphogenetic Swarm Chemistry) was developed using somewhat arbitrary design decisions. The behavioral rules were exactly the same as those used in previous swarm models, which were limited to agents' acceleration only. The state transition functions were defined in a rather simple and linear fashion using a product of a preferential weight matrix and an observation vector. Moreover, the variables included in the observation vector were chosen without substantial justification. We will need to consider adopting more open-ended, nonlinear forms of representations for observations, state transitions and agent behaviors. We plan to apply genetic programming (Banzhaf et al., 2000) or other symbolic evolutionary search methods (Bongard & Lipson, 2007; Schmidt & Lipson, 2009) to overcome this limitation in the future.

Finally, the scope of the experiments was limited only to "free" self-organization of collectives without any stimuli or constraints. Exposing swarms to such external conditions and measuring their adaptive responses will further clarify the functional differences between different classes of morphogenetic collective systems.

Acknowledgments

This material is based upon work supported by the US National Science Foundation under Grant No. 1319152.

References

- Baldassarre, G., Parisi, D., and Nolfi, S. (2006). Distributed coordination of simulated robots based on self-organization. *Artificial Life*, 12: 289–311.
- Banzhaf, W., Koza, J. R., Ryan, C., Spector, L., and Jacob, C. (2000). Genetic programming. *Intelligent Systems and their Applications*, 15(3): 74–84.
- Ben-Jacob, E. (2009). Learning from bacteria about natural information processing. Annals of the New York Academy of Sciences, 1178: 78–90.
- Ben-Jacob, E., Cohen, I., and Gutnick, D. L. (1998). Cooperative organization of bacterial colonies: From genotype to morphotype. *Annual Review of Microbiology*, 52: 779–806.
- Bernoff, A. J. and Topaz, C. M. (2011). A primer of swarm equilibria. *SIAM Journal of Applied Dynamical Systems*, 10: 212–250.
- Beshers, S. N. and Fewell, J. H. (2001). Models of division of labor in social insects. *Annual Review of Entomology*, 46: 413–440.
- Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press.
- Bonabeau, E., Sobkowski, A., Theraulaz, G., and Deneubourg, J.-L. (1997). Adaptive task allocation inspired by a model of division of labor in social insects. In Lundh, D., Olsson, B. and Narayanan A., eds., *Bio Computation and Emergent Computing*, pages 36–45.
- Bongard, J. and Lipson, H. (2007). Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104: 9943–9948.
- Braha, D., Minai, A. A., and Bar-Yam, Y., eds. (2006). *Complex Engineered Systems: Science Meets Technology.* Springer.
- Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, J., Theraulaz G., and Bonabeau, E. (2000). *Self-Organization in Biological Systems*. Princeton University Press.
- Campos, M., Bonabeau, E., Theraulaz, G. and Deneubourg, J.-L. (2000). Dynamic scheduling and division of labor in social insects. *Adaptive Behavior*, 8: 83–95.
- Chate, H., Ginelli, F., Gregoire, G., and Raynaud, F. (2008). Collective motion of self-propelled particles interacting without cohesion. *Physical Review E*, 77: 046113.
- Chuang, Y. L., D'Orsogna, M. R., Marthaler, D., Bertozzi, A. L., and Chayes, L. S. (2007). State transitions and the continuum limit for a 2D interacting, self-propelled particle system. *Physica D*, 232: 33–47.
- Couzin, I. D., Krause, J., Franks, N. R., and Levin, S. A. (2005). Effective leadership and decision making in animal groups on the move. *Nature*, 433: 513–516.
- Couzin, I. D., Krause, J., James, R., Ruxton, G. D., and Franks, N. R. (2002). Collective memory and spatial sorting in animal groups. *Journal of Theoretical Biology*, 218: 1–11.
- Dorigo, M. et al. (2001–2005). Swarm-Bots Project. http://www.swarm-bots.org/.

- D'Orsogna, M. R., Chuang, Y. L., Bertozzi, A. L., and Chayes, L. (2006). Self-propelled particles with soft-core interactions: Patterns, stability, and collapse. *Physical Review Letters*, 96: 104302.
- Doursat, R. (2008). Organically grown architectures: Creating decentralized, autonomous systems by embryomorphic engineering. In Würtz, R. P., ed., *Organic Computing*, pages 167– 200. Springer.
- Doursat, R. (2011). The myriads of Alife: Importing complex systems and self-organization into engineering. In *Proceedings* of the Third IEEE Symposium on Artificial Life, pages 1–8.
- Doursat, R., Sayama, H. and Michel, O., eds. (2012). *Morphogenetic Engineering: Toward Programmable Complex Systems*, Springer.
- Engelbrecht, A. P. (2005). Fundamentals of Computational Swarm Intelligence. John Wiley and Sons.
- Graves, G. R. and Gotelli, N. J. (1993). Assembly of avian mixedspecies flocks in Amazonia. *Proceedings of the National Academy of Sciences*, 90: 1388–1391.
- Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. Proceedings of the 7th Python in Science Conference, pages 11–15.
- Hemelrijk, C. K. and Kunz, H. (2005). Density distribution and size sorting in fish schools: an individual-based model. *Behavioral Ecology*, 16: 178–187.
- Ingham, C. J. and Ben-Jacob, E. (2008). Swarming and complex pattern formation in Paenibacillus vortex studied by imaging and tracking cells. *BMC Microbiology*, 8: 36.
- Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pages 1942–1948.
- Kunz, H. and Hemelrijk, C. K. (2003). Artificial fish schools: Collective effects of school size, body size, and body form. *Artificial Life*, 9: 237–253.
- Leonard, N. and Fiorelli, E. (2001). Virtual leaders, artificial potentials and coordinated control of groups. In *Proceedings of the 40th IEEE Conference on Decision and Control*, pages 2968–2973.
- Mogilner, A. and Edelstein-Keshet, L. (1998). A non-local model of a swarm. *Journal of Mathematical Biology*, 38: 534–570.
- Newman, J. and Sayama, H. (2008). Effect of sensory blind zones on milling behavior in a dynamic self-propelled particle model. *Physical Review E*, 78: 011913.
- Paley, D. A., Leonard, N. E., Sepulchre, R. J., and Couzin, I. D. (2008). Spatial models of bistability in biological collectives. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 4851–4856.
- Parrish, J. K. and Edelstein-Keshet, L. (1999). Complexity, pattern, and evolutionary trade-offs in animal aggregation. *Science*, 284: 99–101.
- Pfeifer, R., Iida, F., and Bongard, J. (2005). New robotics: Design principles for intelligent systems. *Artificial Life*, 11: 99–120.

- Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4): 25–34.
- Romanczuk, P., Couzin, I. D., and Schimansky-Geier, L. (2009). Collective motion due to individual escape and pursuit response. *Physical Review Letters*, 102: 010602.
- Sayama, H. (2009). Swarm chemistry. Artificial Life, 15: 105-114.
- Sayama, H. (2010). Robust morphogenesis of robotic swarms. *IEEE Computational Intelligence Magazine*, 5(3): 43–49.
- Sayama, H. (2011). Seeking open-ended evolution in Swarm Chemistry. *Proceedings of the Third IEEE Symposium on Artificial Life (IEEE ALIFE 2011)*, pages 186–193.
- Sayama, H. (2012a). Morphologies of self-organizing swarms in 3D Swarm Chemistry. *Proceedings of the 2012 Genetic and Evolutionary Computation Conference (GECCO 2012)*, pages 577–584.
- Sayama, H. (2012b). Swarm-based morphogenetic artificial life. In Doursat, R., Sayama, H. and Michel, O., eds., Morphogenetic Engineering: Toward Programmable Complex Systems, pages 191–208.
- Sayama, H. (2013). PyCX: A Python-based simulation code repository for complex systems education. *Complex Adaptive Systems Modeling*, 1: 2.
- Sayama, H. and Wong, C. (2011). Quantifying evolutionary dynamics of Swarm Chemistry. In *Advances in Artificial Life, ECAL 2011: Proceedings of the Eleventh European Conference on Artificial Life,* pages 729–730. MIT Press.
- Schmidt, M. and Lipson, H. (2009). Distilling free-form natural laws from experimental data. *Science*, 324: 81–85.
- Sole, R. and Goodman, B. (2000). Signs of Life: How Complexity Pervades Biology. Basic Books.
- Szabo, B., Szollosi, G. J., Gonci, B., Juranyi, Zs., Selmeczi, D., and Vicsek, T. (2006). Phase transition in the collective migration of tissue cells: experiment and model. *Physical Review E*, 74: 061908.
- Tian, B.-M., Yang, H.-X., Li W., Wang, W.-X., Wang, B.-H., and Zhou, T. (2009). Optimal view angle in collective dynamics of self-propelled agents. *Physical Review E*, 79: 052102.
- Turner, J. S. (2000). *The Extended Organism: The Physiology of Animal-Built Structures*. Harvard University Press.
- Turner, J. S. (2007). The Tinkerer's Accomplice: How Design Emerges from Life Itself. Harvard University Press.
- Turner, J. S. (2011). Termites as models of swarm cognition. *Swarm Intelligence*, 5: 19–43.
- Vicsek, T., Czirok, A., Ben-Jacob, E., Cohen, I., and Shochet, O. (1995). Novel type of phase-transition in a system of selfdriven particles. *Physical Review Letters*, 75: 1226–1229.
- Vicsek, T. and Zafiris, A. (2012). Collective motion. *Physics Reports*, 517: 71–140.
- Wang, X. R., Miller, J. M., Lizier, J. T., Prokopenko, M., and Rossi, L. F. (2012). Quantifying and tracing information cascades in swarms. *PLOS ONE*, 7(7), e40084.

Can active perception generate bistability?

Heterogeneous collective dynamics and vascular patterning

Katie Bentley¹, Kyle I Harrington² and Erzsébet Ravasz Regan¹

¹Beth Israel Deaconess Medical Center, Harvard Medical School, Boston MA
²DEMO Lab, Computer Science Department, Brandeis University, Waltham, MA
kbentley@bidmc.harvard.edu

Abstract

During morphogenesis (the generation of form), biological cells, agents or robots must collectively coordinate where and when to move. How to solve such complex, spatial problems in a timely manner, is fundamental to survival in biological organisms, though temporal regulators are largely unexplored. We take the generation of new blood vessel networks (angiogenesis) as our case study system, where tissues low in oxygen stimulate endothelial cells "ECs" (the inner lining of blood vessels) to grow new network branches. This requires ECs to take on heterogeneous states by collectively competing with one another for migratory status via lateral inhibition.

We propose here that the traditional "decide then move" perspective of cell behavior in angiogenesis may miss a key temporal regulator as it is too slow to account for the rapid, adaptive assignment of heterogeneous cell states. Here we show that a "move and decide" view may provide a better account. In a study focused on an individual EC in a simulated collective, we find that active perception (sensorimotor feedback) can generate bistability through migration-induced cell shape changes. We further exemplify that when parameters affecting active perception are modulated, bistability is lost in the single cell. As a consequence, active perception can directly modulate collective decision timing.

Introduction

Collective behavior during morphogenesis requires timely coordination of many autonomous agents, which becomes increasingly complex if the task requires agents have heterogeneous and adaptive phenotypes. Understanding collective coordination mechanisms holds great promise for morphogenetic engineering of well-adapted robot designs (Doursat et al 2012). To this end, the vasculature in living systems is a perfect case study. New vessels grow and remodel in a dynamic adaptive way to maintain a network with near-intimate contact to every cell in the body, required due to the diffusion limit of Oxygen (Aird 2005). When new vessel growth (angiogenesis) is needed, e.g., in development or wound healing, endothelial cells (ECs) lining blood vessel tubes adaptively respond to the release of diffusing growth factors from hypoxic (low in oxygen) tissue. The ECs then collectively coordinate such that some cells migrate and lead new tubular branches ("sprouts"), while others line the tube walls and ensure that the branches are well spaced (Geudens and Gerhardt 2011) (Fig. 1a). The selection of these two states

- "active" or "inhibited" cell movement - is known to be coordinated by Notch-driven "lateral inhibition", where cells battle to inhibit their neighbors (Hellström et al 2007, Jakobsson et al 2009). Lateral inhibition is known to generate stable alternating patterns of on and off cell states in many systems, hereafter referred to as a Salt and Pepper pattern "S&P pattern" (Collier et al 1996). Recently we showed, via an integrated *in silico* and *in vivo* approach, that cell states are not fixed once selected, but rather dynamically adapt and flip throughout angiogenesis (Bentley et al 2009; Jakobsson et al 2010; Bentley et al 2014a). This occurs when ECs 1) meet new neighbors during branch fusion (required to form

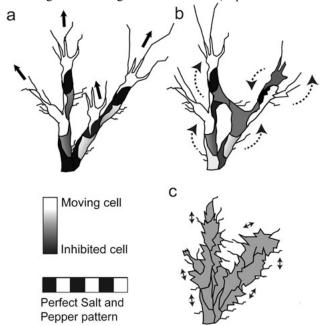


Figure 1. a) Endothelial cells, initially all the same, collectively coordinate to become heterogeneous when stimulated to grow new blood vessel branches. Some cells move, leading new sprouts, others are inhibited, lining the branch and keeping them regularly spaced. The optimal spatial arrangement is a "salt and pepper" (S&P) pattern (alternating states). b) During branching the branches fuse to forming new loops of the network and cells interchange positions. c) If reassignment of states is not rapid the system will fall into homogenous phases as the states are still deciding, disrupting well spaced branching and thickening vessels.

network loops and support blood flow; Fig 1b) (Bentley et al 2009) and 2) rearrange their positions in the collective (Jakobsson et al 2010). Thus, the cellular collective must be capable of rapidly re-establishing the S&P pattern of behaviors in the face of local neighborhood changes, to keep the network branching structure optimized. Otherwise, cells would drift into a "half-way house" state of homogenous movement (Fig. 1c). Lack of differential movement has been shown to disrupt branching as cells hypersprout instead (Hellstrom et al 2008), and has been implicated in abnormal vessel thickening in disease (Bentley et al 2014a). Here we ask, is there an extra dimension of temporal regulation that keeps the system from lingering in a homogeneous state as the process unfolds.

Do cells "Decide then Move" or "Move and decide"?

In Biology, signaling cascades regulating cell behavior are usually presented in a feed-forward manner (Fig. 2a). E.g. ECs are selected as migratory collectively, by Notch lateral inhibition and then those selected will migrate, they "decide then move". However, the relative timing of events indicates that things may, in fact, be more complex.

Lateral inhibition communication between the ECs can be summarized as follows: each cell detects a diffusing stimulatory signal from the hypoxic tissue, primarily Vascular Endothelial Growth Factor (hereafter input signal, "P") by activation of its VEGFR receptors (hereafter sensors, "S"). Sensor activation leads to the up-regulation of the ligand Dll4 ("D"), which then binds to and activates Notch receptors ("N") on neighboring cells. When N is active, the cell downregulates S. Several amplification cycles of this pathway then leads to one cell inhibiting the other's ability to sense, more than the inhibition it, itself, receives (see Figs. 2c-e).

The same sensors also trigger the migratory response of the cell, by locally activating the actin cytoskeleton to first polymerize, generating long thin membrane protrusions called "filopodia" that reach into the environment. Second, actomyosin contractions and adhesion dynamics propel the cell forward. Thus a cell that is more inhibited by Notch signaling, will have less sensors and so will be less migratory (the "inhibited" state).

Lateral inhibition requires gene regulatory changes in the nucleus and protein synthesis, which can take on the order of hours. While faster than multiple cycles required for lateral inhibition, a full migratory response (e.g., where single cells rearrange their positions) also occurs on the time-scale of hours (Jakobsson et al 2010). However, the *initial* migratory response of filopodia growth occurs more rapidly (on the order of minutes) and locally (at the cell surface), without the need to alter gene expression. It is thus logical to assume that the cell shape changes (filopodia) and early movements of the cells membrane during the initial stages of migration will proceed ahead of the established collective decision making process of lateral inhibition.

Consequently, we propose a "move and decide" strategy better captures the process.It is important to note that, the sensors themselves move rapidly during motion, as they reside on the deforming membrane surface. As filopodia extend and retract, sensors are moved to new locations in the environment, continually changing their input (Fig 2c). If the environment contains a gradient, this process has been indicated to speed up collective decision making (Bentley et al 2008). However, the exact reason for this has not been

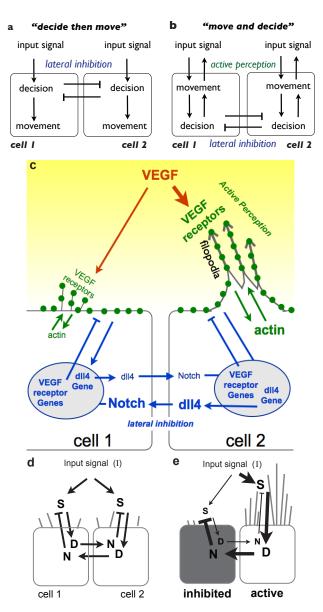


Figure 2. a) traditional feed-forward view of cell behavior regulation in biology termed here a "decide then move" strategy. b) The alternative "move and decide" strategy. c) Signaling pathways involved in blood vessel selection of heterogeneous movement behaviors. Green dots indicate that sensors (VEGFR receptors) reside on the cell membrane, which is deformed and moved as actin polymerizes to form filopodia creating a sensorimotor loop (active perception) d-e) simplified model of (c) S = sensors, D = Dll4, N = Notch, I = input signal. Overtime cells battle back and forth to inhibit each other with one eventually acquiring more filopodia and inhibiting its neighbor more.

established. Here we propose that filopodia act as a means to perform *active perception* of the tissue environment, similar to the proposition that flagella motion confer sensorimotor dynamics and indeed minimal cognition to migrating E. coli cells (Van Duijn et al 2006). While the importance of embodiment in vascular morphogenesis has recently been highlighted (Bentley et. al., 2014b), here we further posit that such active perception confers *bistability* to the active/

inhibited states, rather than the other way around, as currently thought.

An ability to continually adapt and inform our decision making as we move through and experience more of our environment via active perception is known to confer an advantage to decision making, with many studies in robotics and child development e.g. Piaget (1967), Pfeifer (1997) and Beer (2000). As Dewey (1896, pp.137-138) so elegantly put:

We begin not with a sensory stimulus, but with a sensorimotor co-ordination... In a certain sense it is the movement, which is primary, and the sensation which is secondary..."

To establish whether active perception feedback by individual ECs in a "move and decide" strategy may confer robustness and optimize timing of collective decisions, we consider first a tightly controlled, single EC scenario, where collective coordination by lateral inhibition is de-coupled. We hypothesize that active perception alone, prior to any engagement of lateral inhibition feedback, can generate bistability of active/inhibited states. This crucially implies that in situations where single ECs experience intermediate levels of inhibition from their neighbors (common during collective decision making), bistability allows them to pre-commit to an activated or inhibited state. The subsequent lateral inhibition process is thus never called to amplify rapid random fluctuations in D difference between two neighbors. Rather, in each moment each single cell makes a clear decision, and presents its "vote" for lateral inhibition to amplify and lock in, speeding up coordinated decisions.

Bistability, hysteresis and positive feedback

The ability to dynamically select between two *distinct* phenotypes, while intermediate states are unstable, is a hallmark of bistable regulatory systems. In general, bistable systems can maintain two distinct locally stable states in a *single* environment. A simple mechanical analogy is that of a ball found on a double-well landscape (Fig. 3a). In this case, the physical shape of the landscape supporting the ball against gravity creates two stable states, i.e., the bottom of each well.

In the presence of any noise able to overcome friction between ball and surface, this mechanical system will not linger in an intermediate state, at the top of the barrier or between valleys. Positing an iron ball and an external magnet, it is also easy to appreciate that under certain environmental conditions (e.g. magnet on the right side of double-well) this system may be rendered mono-stable. Transitions between the two locally stable states take place abruptly, whenever forces on the ball exceed the potential barrier imposed by the presence of gravity. Moreover, slowly tuning the environment by moving the magnet position left to right and back again (such that the ball is restricted to the left / right / left valleys) uncovers a signature behavior of bistable systems, namely hysteresis (Fig. 3b). Charting the position of the ball as a function of the position of the magnet clearly reveals the initial-condition dependence of the ball's response. For any position of the magnet within a window around the barrier, the ball can settle into either of the two valleys, but which one depends on its recent history of positions. It cannot be stable in the transition state on top of the barrier, it will fall into one

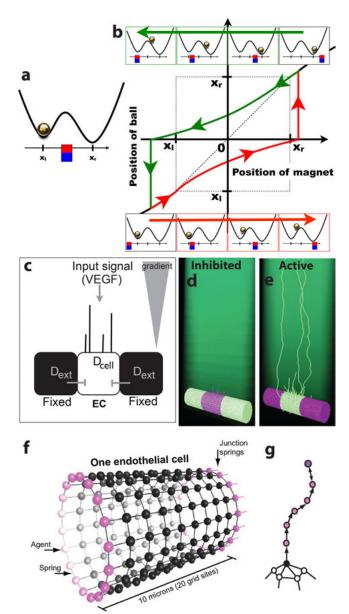


Figure 3. a) A mechanical bistable system - metal ball in a gravitational double-well potential. b) Moving a strong magnet under the ball acts as environmental input, capable of restricting the ball into a single well. As the magnet is slowly shifted from left to right (x-axis), the ball's position (y-axis) first changes slowly (bottom row; red), then undergoes an abrupt jump as the ball clears the barrier. Reverse arm: top row, green. In a wide region of magnetpositions, the ball's dynamics are bistable: it can reside in either valley depending on its initial conditions. c) The APB model setup, we track a single EC's activity (D_{cell}) as it responds to I in the environment and D_{ext} from its fixed neighbor cells. d-e) Frames at t= 300 from APB simulations, D levels high to low shown light green to purple, I gradient in environment shown dark green. d) A run where fixed neighbors have $D_{\text{ext}} = 10000 \, (D_{\text{max}})$ the central EC becomes inhibited (few short filopodia and low D_{cell}). e) A run where $D_{\rm ext} = 0$, the EC becomes active, $D_{\rm cell}$ is high and it has lots of long filopodia. f) The memAgent Model of an EC, reproduced from (Bentley et al 2009). The cell membrane is represented by a

cylindrical, single layer square lattice mesh of agents "memAgents", connected by springs. At either end (adjacent cells not shown) specialized junction springs and agents (pink) connect the cell to its neighbor cells along the vessel. g) To grow a filopodium new memAgents and springs are created.

of the valleys.

How can active perception endow ECs with such bistability? In regulatory circuits, bistability usually arises due to positive feedback (Angeli et.al., 2004; Cherry and Adler, 2000; Gardner et al., 2000; Ferrell, 2002). Active perception through filopodia and cell shape changes that extend the sensors to areas with high levels of input signal generate a potent regulatory drive for further filopodia extension (Fig. 2c). We therefore hypothesize that at intermediate levels of lateral inhibition by neighbors (when collective decisions are still in process) filopodia extensions are difficult enough to initiate that an inhibited state is locally stable. At the same time, should the cell already possess a large number of filopodia extended into a steep input gradient, these send enough signal back to stabilize and maintain an active moving state (Fig. 3g).

The Active Perception Bistability (APB) Model

To test our hypothesis, we developed a new simulation testbed that parallels the above mechanical example. We focus on how an individual cell's state changes as it experiences different, controlled levels of inhibition from two immobilized neighbor cells. Lateral inhibition from these two neighbors ("External D" or $D_{\rm ext}$) affects the individual EC under investigation (its current D level, " $D_{\rm cell}$ ", is used as proxy for its active/inhibited state), but these neighbors are uncoupled from being inhibited themselves (Fig 3c). $D_{\rm cell}$ is therefore analogous to the "ball" in the mechanical example of a bistable system and the fixed neighbors to the magnet; raising $D_{\rm ext}$ can push the cell into a more inhibited (low $D_{\rm cell}$) state and vice versa (Fig. 3d,e).

To simulate this scenario we use the memAgent Spring model (MSM model), which has been previously calibrated to experimental angiogenesis data – for full methods see Bentley et al (2008, 2009). Briefly, in this model cell morphology is represented as a surface comprised of many small agents (membrane agents, "memAgents") connected by springs following Hooke's law (which confer tension to the changing cell shape, akin to the actin cortex beneath a cells membrane) (Fig. 3f). The cell is 10 microns wide with 6 microns diameter (matching in vivo capillary dimensions). The world around it has dimensions (x,y,z=30,58,10 microns). The memAgents move in continuous space, but are snapped to a gridded lattice to implement local rules (grid sites represent $0.5 \times 0.5 \times 0.5$ micron cubes of space).

At each time-step an equal number of the cell's current, total sensors (S_{cell}) are distributed equally to each memAgent (S_{m}) over the cell's current surface shape. Dll4 ligands (D) are equally distributed to any memAgent in Moore neighborhoods of memAgents from other cells (on the interface between cells or "junctions"). The input signal (I) is modeled as a fixed linear gradient, where each grid site G above the vessel has I_G

level, calculated as $I_G = Vy$ where y is the y axis coordinate position of G and V is a tunable constant.

Signaling: Simplifying the description of the competitive receptor biology between different VEGFRs in the original MSM model, as it is not the direct focus of this study, we can state that the number of active sensors of a given memAgent m is calculated as $S'_m = S \cdot \sum I_G$. Notch is activated in each memAgent (resulting in the local removal of D from neighboring cell's memAgents), up to the value of N_m . If D < N then $N'_m = D$; if D > N, then $N' = N_{\text{max}}$.

Filopodia: memAgents extend filopodia by one grid site length at a time by creating non-branched chains of new memAgents adhered to the environment (fixed position) (Fig. 3g). These events are stochastic, with probability:

$$P(\text{extend filopodia}) = F \cdot S',$$

where F is a tunable constant. For full rules behind the actin dynamics, springs and enforcing the chain extension see Bentley et al (2009). Filopodia retract if the memAgent at the top of the chain has not extended for 10 time steps, which is more likely when sensors do not activate. They retract by iteratively snapping the chain of springs back to the next memAgent in the chain, whilst deleting memAgents as they go (based on in vitro observations). The cells have no limit on the length of a single filopodium, or how many they can initiate. However, their total length cannot exceed the total amount of actin available to the cell. The original model in Bentley et al (2009) generates full cell migration by releasing filopodia adhesions and allowing the springs to pull the cell forward. Here, however we keep adhesions fixed to control variables and ask whether shape changes due to filopodia alone can generate bistability.

Genetic Regulation: Briefly, the lateral inhibition is implemented each time step as follows:

$$S_{\text{cell}} = S_{\text{max}} - \sigma N''_{\text{cell}}$$

 $D_{\text{cell}} = D_{\text{cell}} + \delta S''_{\text{cell}}$

where σ , δ are constants and S'' denotes S' after a time delay (28 time steps) corresponding to the time it takes for transcription to alter gene expression in the nucleus, and subsequently protein levels at the cell surface. The new S_{cell} and D_{cell} levels are then redistributed to the memAgents and the process repeats on the next timestep.

Parameterization: All parameters in the model are set as in Bentley et al (2009), which were previously calibrated to experimental data. To investigate active perception we focus here on simulations varying its critical determinants: filopodia extension (F) and the input signal gradient (V). For normal EC behavior F = 2 and V = 0.04.

Results

Hysteresis simulations

We start with D_{ext} set to zero and thus simulate a fully active EC with long filopodia (Fig. 3e). We then slowly raise

the $D_{\rm ext}$ in increments of 50 until full inhibition is achieved, then reverse the process until it is yet again relieved. If active perception indeed creates bistability, we expect the transition between active/inhibited states to take place abruptly, at a $D_{\rm ext}$ threshold that is higher for the active \rightarrow inhibited transition than the reverse. In order to approximate quasi-static equilibrium while changing $D_{\rm ext}$, we tested the stability of $D_{\rm cell}$

levels using the following method: 1) we allowed a transient time window (250 time-steps) to pass, 2) we calculated the average and standard deviation of D_{cell} in two adjacent, non-overlapping sliding windows of size 250, 3) we tested whether $|\mu_c - \mu_p|/(\mu_c + \mu_p) < \varepsilon$ and $|\sigma_c - \sigma_p|/(\sigma_c + \sigma_p) < \varepsilon$, where μ_c/μ_p denote the average D_{cell} and σ_c/σ_p its standard deviation in the last/previous window ($\varepsilon = 0.5$). If the EC does not pass the

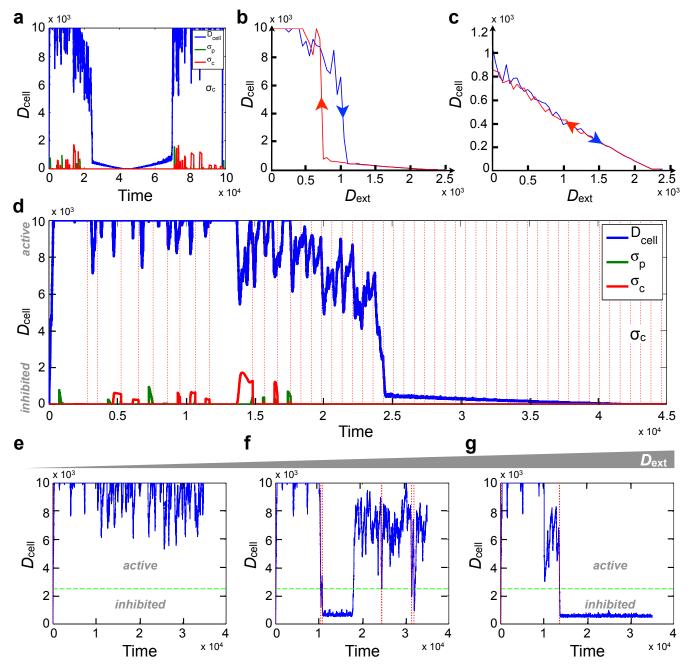


Figure 4. a) Time course of D_{cell} during a full cycle of hysteresis (blue). Standard deviation of D_{cell} , measured in two non-overlapping, expanding windows (σ_p for previous and σ_c for current windows) are shown in green and red, respectively. b) Hysteresis in the active / inhibited transitions of a single EC exposed to slowly increasing (blue), then decreasing (red) D_{ext} . c) Loss of hysteresis in a single EC with weak filipodia extensions, due to reduced sensorimotor feedback. d) Zoomed-in time course of D_{cell} during a half-cycle of hysteresis, showing large D_{cell} fluctuations in the active state. Red lines: times of D_{ext} increment. e-g) Long D_{cell} time-courses of ECs exposed to fixed D_{ext} . Green line: rarest observed D_{cell} value, marking the position of the barrier. Red line: transitions between active/inhibited states. e) $D_{\text{ext}} = 500$; EC restricted to the active state. f) $D_{\text{ext}} = 850$; bistable EC stochastically transitioning between active and inhibited states. g) $D_{\text{ext}} = 950$; EC restricted to its inhibited state.

test or a maximum time limit 60,000 was not yet reached, we increment the window. Otherwise we move on to the next $D_{\rm ext}$. Figure 4a shows the time-dependent evolution of $D_{\rm cell}$ through a full hysteresis cycle, along with its standard deviation measured in each window. As Figure 4b indicates, the system shows hysteresis behavior: as $D_{\rm ext}$ increases (blue arm of hysteresis curve), the cell goes through an abrupt transition from active (high $D_{\rm cell}$) to an inhibited state. Slowly decreasing $D_{\rm ext}$ (red arm) also leads to an abrupt transition from inhibited to active state, but $D_{\rm ext}$ needs to be lowered below the point at which the initial inhibition occurred. This indicates that there is a region of $D_{\rm ext}$ where the EC can exist in either its active or inhibited state, depending on its history.

To prove that active perception is required for this bistability, we repeated the simulations decreasing the rate of filopodia extensions (F) to half their normal value. When the filopodia-dependent sensorimotor feedback is thus weakened, the EC looses its bistability and becomes tunable, in that D_{cell} decreases proportionally with D_{ext} (Fig. 4c).

Due to the stochastic nature of filopodia extensions, D_{cell} fluctuations close to the bistable region are large (Fig. 4d; this is the rationale for the generous ε threshold of 0.5). Due to this inherent stochasticity, the exact D_{ext} thresholds at which abrupt transitions occur are not the same from simulation to simulation. Moreover, if the time window between increments is very large (well above the biological time scales involved in patterning), the EC may undergo stochastic transitions between metastable active/inhibited states. To understand the nature of these transitions, we set up a second set of simulations. In this case, we started the EC with $D_{\text{ext}} = 0$ and ran the simulation for 10^4 time steps, after which we abruptly increased D_{ext} to a nonzero target value, and performed $3 \cdot 10^4$

more steps. Figure 4e indicates that exposure to low $D_{\rm ext}$ (500) keeps the EC in an active state, albeit with sizable $D_{\rm cell}$ fluctuations. At the intermediate $D_{\rm ext}$ level of 850, however, the long simulation showcases abrupt, stochastic transitions between active and inhibited states (Fig. 4f). Lastly, high $D_{\rm ext}$ (950) causes a one-way transition from active to a strongly inhibited state (Fig. 4g). Even here, the transition is not immediate (\sim 0.3·10⁴ steps), but it is irreversible within the time window of the run.

To understand the dependence of the ECs residence time in each state, we partitioned D_{cell} values into active and inhibited

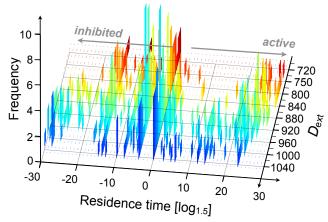


Figure 5. Distribution of residence times in the active and inhibited state (indicated by positive and negative x-values, respectively) in the course of long simulations in the presence of constant $D_{\rm ext}$ (red:low to blue: high). As $D_{\rm ext}$ increases, long residence times in the inhibited state gradually appear, while residence in the active state shortens.

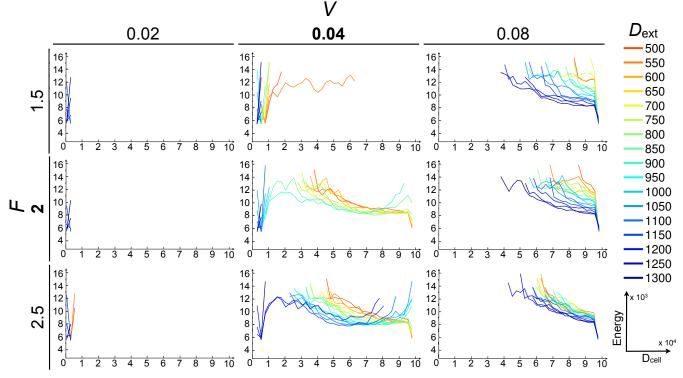


Figure 6. Free energy landscapes of the EC in the presence of constant D_{ext} , ranging from 550 (red) to 1300 (dark blue), as a function of V and F. Energy on the y-axis is calculated as $E = -\log[p(D_{\text{cell}})]$. Bold V/F values (middle panel) mark the control (biologically calibrated) case.

by choosing 2250 as the threshold of transition from Fig. 4d, and measured the length of uninterrupted residence times in each state. Their distribution as a function of $D_{\rm ext}$ is shown in Fig. 5. As expected, an increase in $D_{\rm ext}$ leads to the appearance of long residence in the inhibited state. In the $D_{\rm ext}$ range of 700-1040, the EC is not easily stuck in either state, and often performs rapid back-and forth jumps across the barrier. These rapid jumps are most like in the middle of this $D_{\rm ext}$ window, where long residence in both active and inhibited states are equally likely.

Energy Landscapes

In addition to residence times, long simulations under constant $D_{\rm ext}$ also allow us to map the "free energy landscape" of the EC in different conditions. To this end, we binned the $D_{\rm cell}$ range (B=250), measured the number of nonconsecutive time-steps the EC has $D_{\rm cell}$ in each bin $[n(D_{\rm cell})]$ and calculated the probability distribution

$$p(D_{\text{cell}}) = n(D_{\text{cell}}) / (B \cdot n_{\text{max}})$$

for each target $D_{\rm ext}$ ($n_{\rm max}$ is the total number of time-steps). The free energy of the EC is then proportional to $-\log(p)$. As expected, for intermediate $D_{\rm ext}$ values the energy landscape has two robust local minima (Fig. 6, control case: F=2; V=0.04). We tested the effect of altering active perception on the energy landscape. We find that a double-well energy landscape (and bistability) requires a balance between F and V. For example, altering V has a profound effect within the analyzed range of $D_{\rm ext} \in [550, 1300]$ (values that bracket the range of bistability for the control case). Weak / strong V completely destabilizes the active/inhibited state for this $D_{\rm ext}$ range, and renders the EC monostable. On the other hand F can only destabilize the active state when it is weak. Strong F results in a deeper potential well for the active state, without destabilizing the inhibited valley.

Collective patterning is temporally determined by active perception

So far we have focused on a single EC between fixed neighbors. However, if we have two ECs armed with active perception, they can form individual bistable switches, while D-N lateral inhibition between them guarantees that only one of them can exist in an active state at one time. This circuit design thus guarantees that two neighboring ECs form a higher-level bistable switch. In the setting of a multi-cell collective and in the presence of the input gradient they readily generate the S&P pattern (Fig 7a). In the single cell case, when F is reduced, or V is altered, the active perception and therefore bistability of the cells is weakened. This slows down the collective allocation of heterogeneous migratory phenotype (Fig 7b,c).

Discussion

This study represents a first step towards a new understanding of how coordinated movement decisions in heterogeneous collectives can be made efficient and robust. This paper presents the establishment of a new testbed, the APB model, and validation that the central hypothesis "active perception generates bistability" can be accepted. Hence

active perception and the move and decide perspective provide a novel explanation as to how heterogeneous states are so rapidly and robustly assigned in the collective morphogenesis of blood vessel growth. The bistability of single ECs faced with the intermediate levels of inhibition, typical of unpatterned phases, guarantees that state changes occur abruptly (rapid). Moreover, bistability confers shortterm memory to EC states in the face of small fluctuations (robustness). The sensitivity of angiogenesis to VEGF gradient and concentration changes in disease can now be viewed as an active perception defect. Overall, this study provides novel biological predictions, to be tested in vitro. How full cell movement, collective rearrangement and other signaling pathways feed back and interplay with active perception and collective bistability remains to be investigated in futures studies.

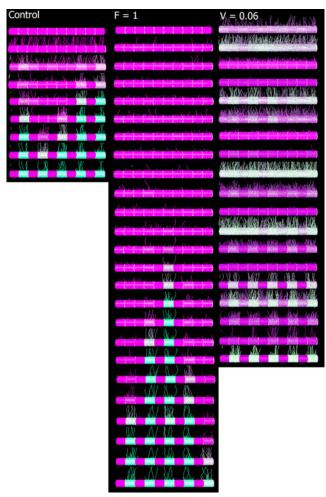


Figure 7. Time-space plots showing simulations with ten cells, periodic boundary conditions and lateral inhibition coupling. Frames every 100 time-steps shown vertically (Dll4 level high-low shown green-purple). It takes approx. 700t to reach the S&P pattern in normal conditions (Control, left panel). Halving the probability of filopodia (F=1, middle panel) slows down the collective patterning by over three times Control. Increasing the input signal (V=0.06, right panel) also slows down collective patterning, but this time due to the cell's synchronously oscillating whilst deciding, as the excessive input causes delayed excessive inhibition.

The reason for the positional interchanges during vascular morphogenesis is not yet known, though the dynamic and plastic nature of the cell phenotypes likely contributes to the adaptive capability of the collective network. This suggests that constant re-allocation of agent behaviors in heterogeneous collective systems via dynamic coordination, such as the active-perception-lateral-inhibition mechanism here, may also confer greater adaptability and robustness to collective robotics. Examples include systems where group plasticity facilitates role coordination, temporary scaffolds, and collective construction (Petersen et al 2011). Thus further study could also reveal important insights for morphogenic engineering principles.

Acknowledgments. KB is funded by BIDMC. ERR is funded by BIDMC and grant HL077348-03 from the NIH. KIH is funded by the Brandeis University Department of Computer Science, and thanks Jordan Pollack for support.

References

- Aird, W. C. "Spatial and temporal dynamics of the endothelium. (2005). *Journal of Thrombosis and Haemostasis* 3(7):1392-1406.
- Angeli D., Ferrell J. E., Sontag E. D. (2004). Detection of multistability, bifurcations, and hysteresis in a large class of biological positive-feedback systems. *Proceedings of the National Academy of Sciences, U.S.A,* 101:1822-1827.
- Beer, R. D. (2000). Dynamical approaches to cognitive science. *Trends in cognitive sciences*, 4(3), 91-99.
- Bentley, K., Gerhardt, H., & Bates, P. A. (2008). Agent-based simulation of notch-mediated tip cell selection in angiogenic sprout initialization. *Journal of Theoretical Biology*, 250(1), 25-36.
- Bentley, K., Mariggi, G., Gerhardt, H., & Bates, P. A. (2009). Tipping the balance: robustness of tip cell selection, migration and fusion in angiogenesis. *PLoS computational biology*, 5(10), e1000549
- Bentley, K., Franco, C. A., Philippides, A., Blanco, R., Dierkes, M., Gebala, V., ... & Gerhardt, H. (2014a). The role of differential VE-cadherin dynamics in cell rearrangement during angiogenesis. *Nature cell biology*, 16:309-321.
- Bentley, K. Philippides, A. and E. Ravasz Regan. (2014b) Do Endothelial Cells Dream of Eclectic Shape? *Dev. Cell.* In Press.
- Cherry J. L., Adler F.R. (2000). How to make a biological switch. *J Theor Biology*, 203:117–133.
- Collier, J. R., Monk, N. A., Maini, P. K., & Lewis, J. H. (1996). Pattern formation by lateral inhibition with feedback: a mathematical model of delta-notch intercellular signaling. *Journal of Theoretical Biology*, 183(4), 429-446
- Dewey, J. (1896) The Reflex Arc Concept in Psychology. *Psychological Review*, *3*, 357-370.
- Doursat, R., Sayama, H., & Michel, O. (2012). Morphogenetic Engineering: Toward Programmable Complex Systems. Springer.
- Ferrell J. E. (2002) Self-perpetuating states in signal transduction: positive feedback, double-negative feedback and

- bistability. Current Opinions in Cell Biology, 14:140 –148.
- Gardner T. S., Cantor C. R., Collins J. J. (2000). Construction of a genetic toggle switch in Escherichia coli. *Nature*, 403:339 –342.
- Geudens, I., & Gerhardt, H. (2011). Coordinating cell behaviour during blood vessel formation. *Development*, 138(21), 4569-4583.
- Hellström, M., Phng, L. K., Hofmann, J. J., Wallgard, E., Coultas, L., Lindblom, P., ... & Betsholtz, C. (2007). Dll4 signalling through Notch1 regulates formation of tip cells during angiogenesis. *Nature*, 445(7129), 776-780.
- Jakobsson, L., Bentley, K., & Gerhardt, H. (2009). VEGFRs and Notch: a dynamic collaboration in vascular patterning. *Biochemical Society Transactions*, *37*(6), 1233.
- Jakobsson, L., Franco, C. A., Bentley, K., Collins, R. T., Ponsioen, B., Aspalter, I. M., ... & Gerhardt, H. (2010). Endothelial cells dynamically compete for the tip cell position during angiogenic sprouting. *Nature cell biology*, *12*(10), 943-953.
- Petersen, K., Nagpal, R., & Werfel, J. (2011). Termes: An autonomous robotic system for three-dimensional collective construction. *Proc. Robotics: Science & Systems VII*.
- Pfeifer, R.a.C.S. (1997). Sensory-Motor Coordination: The Metaphor and Beyond. Robotics and Autonomous Systems *20*, 157--178.
- Piaget, J. (1967). A child's conception of Space (Norton, New York).
- Van Duijn, Marc, Fred Keijzer, and Daan Franken (2006). "Principles of minimal cognition: Casting cognition as sensorimotor coordination." *Adaptive Behavior* 14(2): 157-170.

Inferring Social Structure of Animal Groups From Tracking Data

Brian Hrolenok¹, Hanuma Teja Maddali¹, Michael Novitzky¹, Tucker Balch¹ and Kim Wallen²

¹Georgia Institute of Technology, Atlanta, GA 30332

²Department of Psychology and Yerkes National Primate Research Center, Emory University, Atlanta, GA 30322

Abstract

Inferring the social structures of animal groups from their observed behavior is a non-trivial task usually handled by direct observation. Recent advances in sensing and tracking technology have enabled the collection of dense spatial data over long periods of time automatically. The qualitative differences between sparse hand-coded data and dense tracking data necessitate a new approach to inferring the social structure of the observed animals. We present a framework for using agent-based simulations to guide our approach to inferring social structure from tracking data collected from a small group of rhesus macaques over a period of three months. As part of this framework, we describe a version of the DOM-WORLD model of dominance interactions in rhesus macaques that has been modified to include association preference, and adapted to more closely match the environment where the monkeys were housed. An exploration of simulation results reveals important characteristics of the tracking data. The inferred social structures of the tracked monkeys are also presented.

Introduction

Biologists and psychologists studying the social structures and dynamics of animals have relied on observation by trained researchers for the collection and coding of data, as until recently automated tracking systems have not been able to provide the accuracy required to recognize events of importance. With the advent of new tracking methods and subsequent improvements in tracking accuracy, it is now possible to record accurate, high-frequency spatial information over long periods of time. This qualitatively different kind of data requires a new approach to analysis.

As the sheer volume of data prohibits manual analysis, automated methods are necessary both for identifying key events and inferring relevant characteristics from identified events. In the rest of this paper, we examine how such automated methods can be applied in the specific case of inferring the social structure of a group of six rhesus macaque monkeys given tracking data of their movements over a period of three months at a rate of about 30Hz. In the next section we will review some related literature on social structure and agent-based modeling. Following that we will highlight

the specific aspects of social structure we are interested in recovering, and related behaviors. Next, we lay out our motivation for using agent-based modeling in this work. After that we cover the details of our approach to modeling, and inferring social structure. After that we present some results using simulated and real data. Finally, we provide some high level analysis, conclusions, and directions for future work.

Motivation and Related Work

Social structure in primate groups plays an important role in the health, behavior, and development of group members. Wallen (1996) has shown that social structure plays an important role in the development of behavioral sex differences, while Stephens and Wallen (2013) describe how social status can effect the actual physiological development of young monkeys. Sapolsky (2005) reviews how social status can effect a wide range of health issues, both direct (such as access to resources), and indirect (stress related diseases). Being able to automatically infer the social structure of a group of animals then has wide ranging implications from maintaining the health and safety of laboratory animals, to determining the changes in social structure throughout the course of an experimental protocol.

In order to guide our development of automatic algorithms for inferring social structures from dense tracking data, we take an agent-based modeling approach to creating simulations of animal behavior in order to prototype and refine our methods. The work presented by Yang et al. (2012) has a similar goal, and provides a principled framework for using agent-based models to further the ethological study of foraging behaviors, specifically the foraging behavior of Aphaenogaster cockerelli. Hrolenok and Balch (2013a) presented work on learning these agent-based models of ant foraging directly from data using techniques from machine learning, and later (2013b) fish schooling, although there the purpose was the automated learning of the behavior model itself, while in this work we are interested in developing inference techniques using a known model. Our development of this known model is heavily influenced by DOMWORLD, introduced by Hemelrijk (2000). Hemelrijk

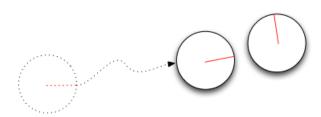


Figure 1: An approaching behavior that indicates an association preference. The strength of the indicated association preference is determined by the frequency and length of periods of close proximity.

presents an agent-based model of dominance in primates that emerges spatial patterns typically found among certain types of rhesus macaque troops.

Social Structures in Rhesus Macaques

One of the most intuitive measures of social structure in primates is association preference, which indicates which members of the group each individual prefers to spend time in close proximity to. A graph constructed from association preferences can illuminate subgroups, key individuals which connect otherwise disconnected groups (also known as cut vertices, or articulation points), as well as overall measures of group structure such as connectedness. The observable behavior where two or more monkeys spend time within relatively close proximity to one another indicates association preference. Figure 1 illustrates a grouping behavior that indicates association.

Another important measure of social structure is the dominance hierarchy. Dominance plays important roles both in interactions between individuals and group dynamics, and changes in dominance can indicate significant events of interest to the primate researcher. Observed displacement and withdrawing behaviors such as chasing and fleeing indicate a dominance relationship. Figure 2 illustrates a withdrawal behavior that indicates a dominance relationship.

While some association, displacement, or withdrawing behaviors can only be identified visually, a large number of them can be detected directly from spatial data, as described in later sections. In order to obtain this data, we utilized a 3D position tracking system to track the positions of 6 monkeys in a 3m x 3m enclosure over a period of three months. Details of the tracking system are described in Huang et al. (2012).

The Importance of Agent Based Models

Agent-based modeling and simulation of animals solve two major problems in the experimental study of animal behavior. First, the data collection cost associated with studies

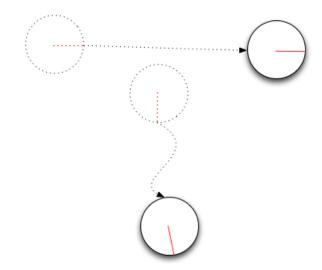


Figure 2: A withdrawal behavior that indicates an dominance relationship. The strength of the indicated dominance relationship is determined by the relative frequency with which each individual withdraws from the other.

done in simulation using high fidelity models is essentially zero, at least compared to the cost of running experiments and collecting data on real animal subjects. Using ABMs allows the researcher to run simulated experiments to increase the confidence of statistical analysis which might otherwise be less conclusive.

Second, when inferring model parameters directly from data, one is faced with the task of validation without access to any "ground truth". Performing the same inference methods on simulated data can provide crucial insight into how those techniques may perform on data from live animals. Both success and failure can be valuable clues into the capabilities and limitations of the inference methods.

In this work, we focus on using ABMs as a validation tool. In the next section we introduce an agent-based model of social interactions between monkeys based on a well studied simulation with slight modifications relevant to the specific social measures mentioned previously. Using this ABM, we can measure quantitatively the effectiveness of our methods for recovering social structure.

Methodology

To validate our method for inferring social structure, we created an agent-based model that incorporates the important behaviors mentioned previously, parameterized in a way that allowed us to compare the recovered social structures with the "ground truth" of the simulation.

Agent Based Behavior Model

Our simulation model, which we call SMALLDOMWORLD is a modification of the earlier DOMWORLD of Hemelrijk

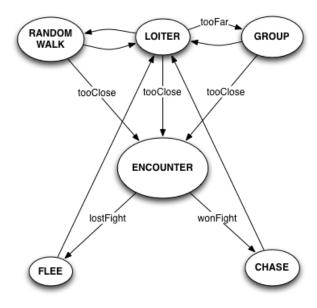


Figure 3: The SMALLDOMWORLD model.

(2000). The behavior of the individuals is guided by three components: a grouping component that draws individuals together, a dominance component where individuals confront each other and the winner chases the loser, and a random component where individuals wander about their environment at low speed.

In order to match the environmental conditions of the animals being studied, we modified the DOMWORLD model presented in Hemelrijk (2000) in a number of ways. In the troop we studied, the dominance relationships were already stable and established, where as in DOMWORLD, dominance relationships are recalculated after every encounter. In DOMWORLD, as in our model, dominance encounters only occur when an individual approaches another within some distance threshold representing an intrusion into personal space. In our model, the probability of an intrusion on personal space resulting in a dominance encounter is given by the parameter σ where $\sigma = 1.0$ indicates a completely stable dominance structure with no confrontations, and $\sigma = 0.0$ ensures that any intrusion results in a confrontation. We use the same dominance confrontation mechanism as DOM-WORLD: each individual is given a dominance weight, and the difference in weights probabilistically determines the winner of any encounter (see Hemelrijk (2000) for details).

We also introduced some selectivity into the grouping behavior. Grouping in DOMWORLD represented a desire by all individuals in the group to remain within some proximity of other group members, and so when an individual found itself far away from the center of the group, it selected another visible member of the group uniformly randomly to head to-

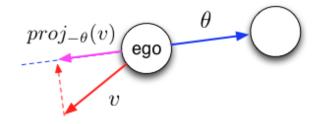


Figure 4: Detection of fleeing events.

wards. In our model, each individual has a list of association preferences $\pi = \langle \pi_1, \pi_2, \dots, \pi_n \rangle$ which can be thought of as the distribution over individuals selected for grouping. This leads to patterns of association which are non-uniform and give rise to the social structures described earlier. The list of association preferences can be combined into a single association preference matrix P with each row corresponding to a single individual's association preferences.

While these two modifications represent the most important changes between our model and DOMWORLD, we also made several changes to accommodate differences in the modeled environment and simulation engine. DOMWORLD's environment is long-range, discrete, and unbounded (toroidal in implementation), whereas our environment is quite small, continuous, and interactions with the boundary of the enclosure are common (which necessitated some kind of obstacle avoidance behaviors). Figure 3 gives a graphical representation of the behavior model of individuals in SMALLDOMWORLD.

Heuristic Behavior Recognition

In this section we present two heuristic methods for identifying association and dominance behaviors, and how they can be used to infer the social structures of a group of monkeys.

Time spent within proximity is a straightforward way to detect behavior which indicates association preference among group members. By counting the frequency and length of events where the ego — by which we mean the individual whose preferences we are trying to determine — comes within a threshold distance of another individual, and remains there at low to zero velocity for at least some minimum period of time, we can infer which individuals the ego prefers to spend time with. If we denote by E_{ij} the time monkeys i and j spend near each other, then we can fill out the entries of the association preference matrix P as:

$$P_{ij} = \frac{E_{ij}}{\sum_{k} E_{ik}} \tag{1}$$

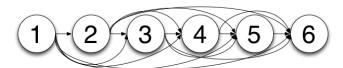


Figure 6: Dominance hierarchy.

While detecting all types of withdrawal events may be difficult, we can capture a certain subset fairly easily. One type of withdrawal involves the ego rapid moving directly away from the target, which we will call *fleeing* events. We can detect fleeing events by counting the length and number of events where the magnitude of the ego's velocity (v) projected onto the bearing (θ) between the ego and target is larger than a threshold (f), which is shown graphically in Figure 4. For each pair of individuals we can compute a dominance measure d as

$$d_{AB} = \left| \left\{ e_{AB} \mid proj_{-\theta(A,B)}(v_B) > f \right\} \right| \tag{2}$$

If individual A flees from individual B more frequently than the opposite $(d_{BA} > d_{AB})$, we can infer that A is subordinate to B. In practice, picking f to correspond to roughly 30 degrees on either side of moving directly away from the target worked well.

Experiments

We performed four experiments to test our approach, three using data collected from our simulation model SMALL-DOMWORLD with different parameterizations of the social structure, and one using the three months of tracking data we collected from a small group of animals. Our purpose in performing the simulation experiments was to measure how accurately we would be able to recover social structure, and to characterize scenarios where our method might not be able to recover social structure. In each of the simulation experiments we ran ten experiments under the same parameterization but with different initial configurations and random seeds.

In the first experiment, we simulated a group of monkeys which had a social structure with two disconnected subgroups, as shown in Figure 5. The parameterization which realized this structure had each individual's association preference set to 1.0 for other members of its subgroup, and 0.0 otherwise. This way there should be no deliberate preference to spend time in proximity of non-subgroup members. The dominance relationships for this and the two following simulation experiments was a direct linear relationship with rank corresponding to ID, as illustrated in Figure 6. The parameterization that realized this model set the dominance weight for the least dominant individual to 1.0, with each individual higher in the chain having twice the dominance

Table 1: Simulation parameters common for all experiments.

Personal distance	0.25m
Near distance	0.8m
Fleeing speed	2.0m/s
Chasing speed	1.0m/s
Grouping speed	0.25m/s
Wander speed	0.12m/s

Table 2: Frobenious error of recovered association preference as compared to a randomly generated symmetric, normalized matrix with zero diagonal. Averaged over 10 runs.

Recovered AP	Avg. error (std.)	random AP	
disconnected	0.1744 (0.0014)	0.2408 (0.0326)	
neutral hinge	0.1002 (0.0015)	0.1797 (0.0350)	
preferred hinge	0.1388 (0.0004)	0.1869 (0.0158)	

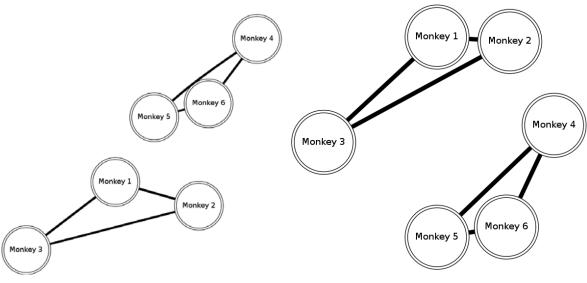
weight as the next lowest, or $(D=2^{N-i})$. The same dominance weights were used in each experiment. Hierarchy stability was set fairly high $(\sigma=0.8)$, so that dominance interactions were not frequent, but still frequent enough to reliably detect the dominance hierarchy. Other simulation parameters are listed in Table 1 and were estimated from tracking data of live monkeys where appropriate, or taken from the literature when available.

We were successfully able to consistently recover both the dominance relationships and association preferences in this experiment. In order to get a sense of our accuracy, we compared the recovered association preference matrix with the ground truth parameterization and with a randomly generated matrix which was restricted to the same form (rownormalized, zero-diagonal, symmetric). The results of this comparison are given in the first row of Table 2, which shows that our recovered parameters are significantly closer to the ground truth than random. In order to recover the graph structure shown in Figure 5, we chose a threshold τ , such that edges larger than τ are included in the graph while those smaller are not. In order to characterize our ability to pick τ reliably, we examined the distribution of values in the association preference matrix P, shown in Figure 7. The distribution is clearly divided into two modes, which indicates that by picking a a threshold between the two modes, our recovered graph will be stable to noise in the estimation of association preferences. In our testing, picking

$$\tau = \frac{1}{n^2} \sum_{i,j} P_{ij} \tag{3}$$

where n is the number of agents, worked reliably.

In the second experiment, we modified the association preferences so that one individual, which we will call the



(a) Recovered association preference.

(b) Ground truth association preference.

Figure 5: Association preferences for the disconnected scenario. The recovered graph (5a) closely matches the actual association preferences used in the simulation (5b). Line thickness corresponds to strength of association preference. Association preferences that fall below the threshold τ (from Equation 3) are not shown.

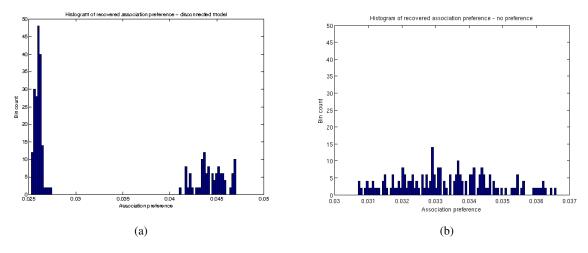
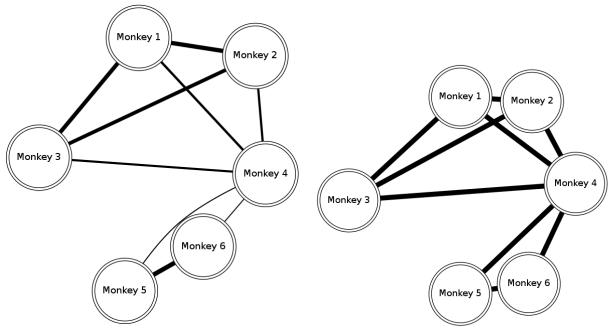


Figure 7: Histogram of association preference values recovered (7a) from the disconnected scenario, and (7b) a simulation with no association preferences. In the second simulation, agents followed the same behavior model, except when choosing to group where they chose among neighbors without preference. Notice the clear separation into two modes of the recovered association preferences as compared to the noisy unimodal distribution from the simulation with no preferences.



(a) Recovered association preference.

(b) Ground truth association preference.

Figure 8: Association preferences with hinge node. The recovered graph (8a) closely matches the actual association preferences used in the simulation (8b). Line thickness corresponds to strength of association preference. Association preferences that fall below the threshold τ are not shown.

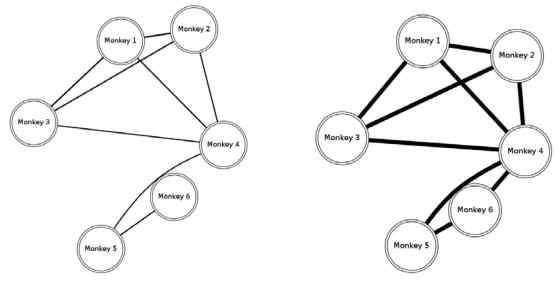
hinge, became an articulation point linking the two subgroups, as shown in Figure 8. To do this, we modified the association parameters such that the hinge individual preferred everyone equally, but no one had a preference for them. In terms of our parameterization, we set the hinge individual's row $P_{hj}=1.0, \forall j,$ and its column $P_{ih}=0.0, \forall i.$ Results are shown in the second row of Figure 2. Again, the recovered association preference is significantly closer to the ground truth than a random association preference, and the dominance hierarchy was recovered without error.

In the third experiment, we repeated the previous experiment, but also allowed the other individuals to preferentially group with the hinge individual by setting $P_{ih} = 1.0, \forall i$. By doing this we highlight a potential scenario where our method may not be able to recover the social structure accurately, specifically non-transitive association preferences. Note that our metric for association preference makes no distinction between individuals which are within proximity because they chose to be, and those that just happen to be nearby. For example, if individuals A and B do not have any preference for association, but each has a high preference for associating with a third party C, then regardless of C's preferences, A and B will spend a high proportion of time in proximity of one another. Figure 9 as well as the third row of Figure 2 illustrate how recovery performance is degraded in this type of non-transitive scenario.

Finally, for our fourth experiment we applied our methods to tracking data of live animals. Figures 10 and 11 show the recovered association preferences and dominance hierarchy for the entire period the monkeys were tracked. We picked τ using the same approach described above, although from examining the distribution of association preferences shown in Figure 12, we know that this choice will be less stable with respect to noise. That is, it is more likely that some edges will be included or excluded from the graph due to small changes in association preference. The dominance relationship is a linear hierarchy (4,3,5,2,6,1) with individual 4 being the most dominant, and individual 1 being the least dominant. This agrees with the recovered association preference, where individuals are shown as preferring to associate with other individuals at similar ranks in the hierarchy.

Conclusion

We have described a framework for using agent-based models to explore the characteristics of automated techniques for analyzing dense spatial data. In the specific case of inferring the social structure of rhesus macaques from tracking data, we have illustrated under what conditions simple heuristic analysis can provide accurate reconstructions of social structure, and provided some insight into the stability and reliability of our approach using an extension of the DOM-WORLD agent-based model to guide our analysis.



(a) Recovered association preference.

(b) Ground truth association preference.

Figure 9: Association preferences for the hinge node scenario using non-transitive preferences. The recovered graph (9a) is missing a link between monkey 6 and monkey 4 in the actual association preference graph (9b). Notice also that the magnitude of the preferences — shown by the thickness of the edges — is much closer to the threshold value τ . Picking smaller τ results in additional edges that are not present in the simulated behavior. The non-transitive preferences make it difficult to choose a stable τ .

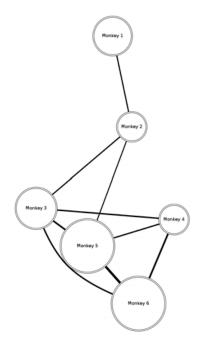


Figure 10: Association preferences for live animals. Diameter of the node is determined by the sum of association preferences for that node. Links are included if they are larger than the mean association preference, and their width is determined by how strong the association preference between the two nodes is.

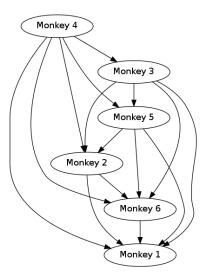


Figure 11: Dominance hierarchy for live animals. Linear chain hierarchies match with our simulated model of dominance behavior, but it is important to note that no part of our inference of dominance relationships *enforces* linear chains. So if the live animals had been an egalitarian troop with little to no aggressive displays, or if the dominance rankings had not been established, we would expect to see a different kind of dominance structure.

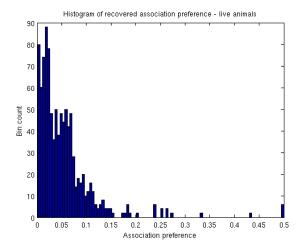


Figure 12: Histogram of association preference values recovered from the live animals. The secondary mode in this distribution is less distinct, but there still is a separation into low and high preference levels.

In future work we would like to expand our approach to include more sophisticated inference techniques. It may be the case that fairly simple probabilistic models will be able to capture some of the structure that we are not able to recover directly from observed behaviors. Association and dominance are clearly not independent relationships, and incorporating some notion of how each affects the other may allow us to improve the accuracy of our inference.

Acknowledgements

This research was supported by the National Institute for Mental Health (MH050268-14S1) as well as by the National Center for Research Resources to the Yerkes National Research Center (P51 RR00165; YNRC Base grant), which is currently supported by the Office of Research Infrastructure Programs/OD P51OD11132. The YNPRC is fully accredited by Americans for the Assessment and Accreditation of Laboratory Care, International.

References

Hemelrijk, C. K. (2000). Towards the integration of social dominance and spatial structure. *Animal Behaviour*, 59(5):1035–1048.

Hrolenok, B. and Balch, T. (2013a). Learning executable models of multiagent behavior from live animal observation. In ICML 2013 Workshop on Machine Learning For System Identification.

Hrolenok, B. and Balch, T. (2013b). Learning schooling behavior from observation. In *Advances in Artificial Life*, ECAL, volume 12, page 686691.

Huang, P., Sawhney, R., Walker, D., Wallen, K., Bobick, A., Qin, S., and Balch, T. (2012). Learning a projective mapping to lo-

cate animals in video using rfid. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3830–3836. IEEE.

Sapolsky, R. M. (2005). The influence of social hierarchy on primate health. *Science*, 308(5722):648–652.

Stephens, S. B. and Wallen, K. (2013). Environmental and social influences on neuroendocrine puberty and behavior in macaques and other nonhuman primates. *Hormones and behavior*, 64(2):226–239.

Wallen, K. (1996). Nature needs nurture: the interaction of hormonal and social influences on the development of behavioral sex differences in rhesus monkeys. *Hormones and Behavior*, 30(4):364–378.

Yang, Y.-T., Quitmeyer, A., Hrolenok, B., Shang, H., Nguyen,
D. B., Balch, T. R., Medina, T., Sherer, C., and Hybinette,
M. (2012). Ant hunt: Towards a validated model of live ant hunting behavior. In *FLAIRS Conference*.

Evolution of Collective Behaviors by Minimizing Surprise

Heiko Hamann

Department of Computer Science, University of Paderborn, Germany heiko.hamann@uni-paderborn.de

Abstract

Similarly to evolving controllers for single robots also controllers for groups of robots can be generated by applying evolutionary algorithms. Usually a fitness function rewards desired behavioral features. Here we investigate an alternative method that generates collective behaviors almost only as a by-product. We roughly follow the idea of Helmholtz that perception is a process based on probabilistic inference and evolve an internal model that is supposed to predict the agent's future perceptions. Separated from this prediction model the agent also evolves a regular controller. Direct selective pressure, however, is only effective on the prediction model by minimizing prediction error (surprise). Our results show that a number of basic collective behaviors emerge by this approach, such as dispersion, aggregation, and flocking. The probability that a certain behavior emerges and also the difficulty of making correct predictions depends on the swarm density. The reported method has potential to be another simple approach to open-ended evolution analogical to the search for novelty.

Introduction

The creation of control algorithms for self-organizing, artificial collective systems is challenging because it is difficult to anticipate how a multitude of local interactions will effect the global behavior. Hence, evolutionary algorithms are a good option. A desired behavior is evolved by defining a fitness function that rewards the occurrence of particular behavioral features. Alternatively, an indirect selective pressure can be generated that does not directly influence the evolved behavior while interesting behaviors should still emerge. In this paper we investigate one of such alternatives by evolving an agent, that tries to predict its future perception, while a number of collective behaviors is obtained almost only as a by-product.

Evolution of collective behaviors

The application of swarm intelligence (Bonabeau et al., 1999) to the field of robotics is called swarm robotics (Dorigo and Şahin, 2004; Brambilla et al., 2013). The problem of designing algorithms for swarm robotics that generate the desired behaviors (swarm engineering,

Brambilla et al. (2013)) is challenging because it means to design self-organizing systems. Self-organization relies on feedback processes and a multitude of interactions (Bonabeau et al., 1999) that have effects which are difficult to anticipate. One solution is to develop global models that predict the expected collective behavior and give insights about underlying principles (Hamann, 2010; Martinoli et al., 2004). Another approach is to apply methods from evolutionary robotics (Nolfi and Floreano, 2000) to swarm robotics, that is, evolutionary swarm robotics (Trianni, 2008). For example, the evolution of an aggregation behavior in robot simulations (Trianni et al., 2003) and also the evolution of communication in combination with collective motion were reported (Trianni et al., 2004).

In evolutionary robotics the design of fitness functions is a challenging issue. For example, the unfavorable approach of defining elaborated complex fitness functions, that predefine many behavioral features of the expected behavior, is feasible. That way intrinsically complex robot tasks are simplified by exploiting a priori knowledge which foils the key idea of evolutionary robotics that it is a preferential approach whenever there is only little a priori knowledge (Nelson et al., 2009). Another issue of fitness functions is that the evolutionary algorithm might converge prematurely. To overcome these challenges, new approaches were reported, such as novelty search and related methods (Lehman and Stanley, 2008; Mouret and Doncieux, 2009). The successful application of novelty search to swarm robotics was recently reported (Gomes et al., 2013). Artificial life is a related field in this context, there the evolution of collective behaviors within artificial ecologies was reported that either rely on selective pressure generated by ecological features only (Schmickl and Crailsheim, 2006) or in combination with explicit fitness functions (Ward et al., 2001).

Behavior based on minimization of surprise

Many common approaches in the field of intelligent agents focus on defining condition-action rules or similar methods to generate intelligent behavior. In evolutionary robotics (Nolfi and Floreano, 2000) the common approach is based on reactive control possibly combined with simple internal states which means complex world models are usually not considered. That leaves a considerable gap to mammals but also to insects such as honeybees that have capabilities for learning and memory (Hammer and Menzel, 1995). An initial step towards the evolution of simple world models would prepare to go beyond the limited capabilities of reactive agents.

There are many theories that try to define a systemic concept of brains which would allow to explain the variety of behaviors we observe in animals. One of these theories goes back to von Helmholtz (1867) who argued that perception is a process based on probabilistic inference. According to that the main challenge imposed on each animal's brain is to determine coherent causes to its sensory inputs. This abstract concept was picked up, for example, in machine learning to solve the unsupervised learning task of creating a model for a given data set. The so-called Helmholtz machines (Dayan et al., 1995) are trained to serve as a generative model, that is, they learn a probabilistic model of an assumed hidden structure of the input data. Following this concept the brain can be understood as a 'prediction machine' that learns models which describe the causes of its perceptions. Furthermore, a plausible assumption to make is that minimizing the prediction error of this model is advantageous for the considered organism. This line of thought is implemented in a mathematical framework by Friston (2010) which defines an information-theoretic analogon to thermodynamic (Helmholtz) free energy. Basically he defines this free energy as the prediction error. This approach based on a 'freeenergy principle' has the potential to be a unified brain theory (Friston, 2010; Friston et al., 2006). The connection to Darwinian evolution is established by the argument that minimal prediction errors indirectly confine the set of stateaction pairs, that are experienced and executed by an agent, which in turn limits the living space and is life prolonging: "By sampling or navigating the environment selectively they restrict their exchange with it within bounds that preserve their physical integrity and allow them to last longer" (Friston et al., 2006). A minimal prediction error is hence interpreted as an evolutionary advantage. While Friston mostly focuses on simple agent–environment interactions, we apply this concept to agent-agent interactions in a collective system here. In the case of a swarm, the future perceptions of an agent not only depend on its own actions (and a possibly dynamic environment) but also on the actions of other close-by agents. This adds another feedback loop to Friston's framework that indirectly asks the prediction machine to predict other prediction machines. These prediction machines can be interpreted as world models as used in intelligent agents. However, they are not applied as a mere tool to make intelligent decisions but as the main driving force for the emergence of intelligent behavior.

Similar work is, for example, that of Capdepuy et al. (2007) which is based on an information-theoretic measure called 'empowerment'. Empowerment is the "maximum potential information an agent can transfer into its own sensors through the environment" (Capdepuy et al., 2007). The reported simulations are of large-scale (1000 agents) in a grid world and mostly aggregation phenomena are observed. Also the work of Oudeyer et al. (2007) is related which focuses, however, on single robot settings. They propose a sophisticated learning approach which tries to put the robot into situations that allow for a maximal learning progress. The derivative of the error rate curve of a learning machine is monitored and a classification system needs to be trained that structures the robot's sensory contexts. Furthermore note that making this connection between the neurosciences and swarm intelligence follows the idea of 'swarm cognition' (Trianni et al., 2010).

Basic collective behaviors

In general there is a big variety of collective behaviors including rather complex behaviors, such as foraging, sorting, nest building, and cooperative transport (Bonabeau et al., 1999). At the other end of the scale there are simple collective behaviors that are restricted to agent–agent interactions without explicit agent–environment interactions. These simple behaviors rely exclusively on agent positions and agent motion while not including environmental features. There are four basic collective behaviors that are based on motion (moving or stopped) and relative positions (minimal distances between agents or maximal distances) only. On the basis of these two dimensions they are categorized: dispersion (maximal distances, stopped), aggregation (minimal distances, stopped), random (maximal distances, moving), and flocking (minimal distances, moving).

In the following we focus on these four collective behaviors of low complexity and investigate the necessary conditions that allow them to emerge. Following the motivation of the free-energy principle, our objective is to provoke the emergence of these behaviors without an explicit external force acting on agent behaviors or agent controllers and also without explicit selective pressure due to ecological features such as ecological niches or co-evolution. The investigated hypothesis is that a mere selection for minimal surprise (i.e., minimal prediction error, minimal free energy, Friston (2010)) is sufficient to evolve these four simple collective behaviors. Note that a short overview of this research was published before (Hamann, 2014).

Model

Our swarm model is roughly inspired by the desert locust, *Schistocerca gregaria*, that shows collective motion (often called 'marching bands') in the growth stage of a wingless nymph (Buhl et al., 2006). Buhl et al. reduce the collective motion of the locusts to a quasi-one-dimensional sys-

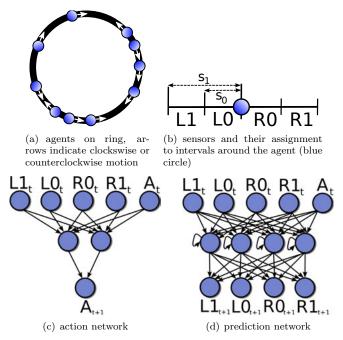


Figure 1: Experimental setting, sensor setting, and artificial neural networks for control and prediction.

tem in their experiments by placing the animals in a ringshaped arena. The observed directional alignment of a majority of locusts is density-dependent and individuals seem to change their direction as a response to neighbors. Here the agents move in 1-d space in the form of a circle, in the following called ring (see Fig. 1a). The ring's circumference is denoted by ring length L. The agents choose from only two available actions: moving clockwise or moving counterclockwise. However, the agents have no global reference frame, that is, whether they are moving clockwise or counterclockwise is unknown to them. Hence, they actually choose from two actions: staying with the current direction or switching the direction. They are not allowed to stop. The

parameter	value
swarm size N (num. agents in simulation)	20
agent sensor range s_0	0.5
agent sensor range s_1	1.0
agent speed v	0.1
noise (displacement of agent position)	[-0.01, 0.01]
evaluation length in time steps T	500
population size (evolution)	50
number of generations	30
num. of sim. runs per fitness evaluation	10
elitism	1
mutation rate	0.05
ring length L (circumference)	[5, 50]

Table 1: Parameter settings.

agents have four sensors (L1,L0,R0,R1) that cover four intervals of their neighborhood defined by sensor distances s_0 and s_1 (see Fig. 1b and table 1 for the used parameters). The sensors are discrete and output either a '1' if there is at least one neighboring agent within the respective interval or a '0' if there is no agent.

In the following experiments the swarm size is fixed to N=20 agents while the ring length L is varied between experiments. This allows to test different swarm densities N/L. Initially the agents are positioned in space by sampling from a random uniform distribution and also their initial direction is chosen with equal probabilities. Time is discrete and each agent's position is updated by adding or subtracting the agent speed of v=0.1. In addition, a small noise is applied to the agent's position by adding a random uniform value sampled from [-0.01, 0.01]. This small noise was added to the system based on the heuristic experience that it triggers a bigger diversity of behaviors. The agents are allowed to pass each other without any interference.

Each agent is equipped with two independent artificial neural networks (ANN) that we call 'action network' and 'prediction network'. The action network has 5 input neurons, 2 hidden neurons, and 1 output neuron (see Fig. 1c). The inputs are the 4 sensor values (L1,L0,R0,R1) and the last action. That way the ANN is a recurrent network although it is implemented as a feedforward network. The output neuron determines the next action (stay with current direction or switch direction) based on a threshold. The prediction network has 5 input neurons, 4 hidden neurons, and 4 output neurons (see Fig. 1d). The input is the same as with the action network. Each hidden neuron has a self-loop which explicitly implements a recurrent network. The output of the prediction network are 4 values that are associated with the 4 intervals of the 4 sensors (L1,L0,R0,R1). Each output neuron determines the predicted value of the respective sensor for the next time step. The agent actually tries to predict where its neighbors will be positioned in the next time step.

We apply a simple evolutionary algorithm. The genomes consist of two sets of weights, one for the action network and one for the prediction network. Note that by applying an evolutionary algorithm we introduce a second concept of a population in this work which is the population of genomes (in difference to the population of agents that are run in the simulations). This population size is 50. Initially a population of random networks is generated (weights are random uniform over [-0.5, 0.5]). The genomes are evaluated by applying the above described swarm simulation. For a particular evaluation run all agents of the simulation share the same genetic material, that is, they have the same networks as defined by the genome that is currently evaluated. The fitness function rewards good predictions of the prediction

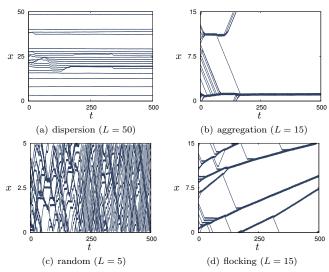


Figure 2: Trajectories of all agents for the four basic swarm behaviors: dispersion, aggregation, random, and flocking.

network. It is defined by

$$F = \frac{1}{NT} \sum_{t \in \{0,1,\dots,T-1\}} \sum_{i \in \{0,1,\dots,N-1\}} \sum_{j \in \{L1,L0,R0,R1\}} c_{i,j}(t)$$
(1)

whereas N is the swarm size, T is the length of the evaluation in time steps, and $c_{i,j}(t)=1$ if the prediction of the previous time step for sensor j of agent i matches the current value of sensor j, otherwise $c_{i,j}(t)=0$. For time t=0 the predictions are set to 0 by definition. The evaluation of a genome is averaged over the results from 10 independent simulation runs. The theoretical best fitness is 4. Note that the behavior of the agents as defined by the output of the action network is not a direct subject to the fitness function because only correct predictions of the prediction network are rewarded. Each weight of both the action network and the prediction network is mutated with a probability of 0.05. Proportionate selection and elitism of 1 are used. Evolution is run for 30 generations totaling to 1500 evaluations (each based on 10 independent runs of the simulator).

Results

All four of the above mentioned basic collective behaviors are found among the best evolved individuals over different swarm densities. Fig. 2 shows example trajectories of all N=20 agents for each of these behaviors (vertical axis gives position x of agents on the ring, horizontal axis gives evolution in time t). The agents cannot simply stop but by permanently switching between clockwise and counterclockwise motion (output of action network always 'switch') they cover no distance averaged over several steps as seen in Fig. 2a. Arguably this behavior can be called dispersion although the agents are initially positioned by sampling from a uniform distribution. Agents that

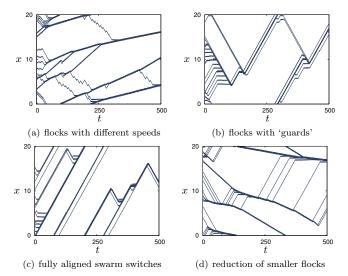


Figure 3: Trajectories of all agents for interesting collective behaviors (L=20).

are initially close-by actually increase the distance between them. Similarly in some runs the agents also manage to stay together in a cluster of high density after they have aggregated as seen in Fig. 2b. Another behavior is seen in Fig. 2c. By rarely switching the direction (output of action network mostly 'stay') the agents cover maximal distance and logically form two groups (clockwise and counterclockwise) distributed over the whole ring. If the agents aggregate but still cover some distance, then we get a flocking behavior as seen in Fig. 2d. Also note that these flocks manage to travel with different velocities (i.e., different cyclic combinations of clockwise/counterclockwise motion). Fig. 3 shows a few more examples of the obtained collective behaviors for a medium swarm density (L = 20) to document the variety of behaviors. Fig. 3a shows an example for flocks moving with different velocities depending on their size. All flocks have positive velocities and small flocks move faster. That way the small flocks catch up and join the bigger flocks. Fig. 3b shows a flock with one agent at each side virtually as 'guards'. Fig. 3c shows a fully aligned swarm that abruptly switches its direction (similarly to observations of locusts, see Buhl et al. (2006). Fig. 3d shows a technique to reduce the size of smaller flocks one by one to form a single flock.

Fitness of best evolved individuals

Next we investigate a series of experiments for varied ring lengths $L \in \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$. Varying the ring length means varying the swarm density because the swarm size is fixed to N=20. For each ring length L, 200 independent evolutionary runs were done. In the following the focus is on the best evolved individuals, that is, the best prediction machines. For the example of ring length L=25 the increase of the best fitness over generations is shown in Fig. 4a (reported before, Hamann (2014)).

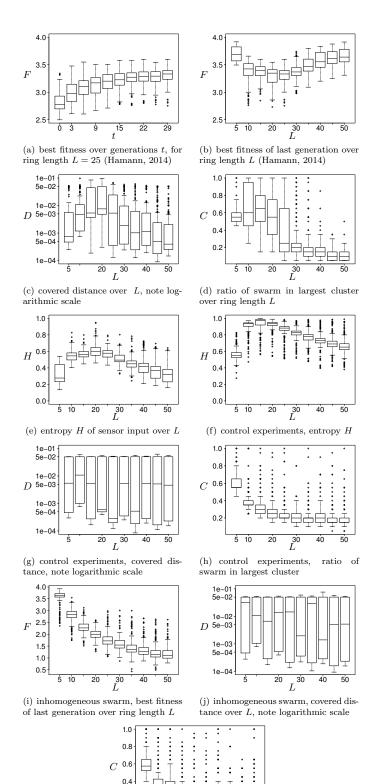


Figure 4: Results – fitness, cov. dist., largest cluster, entropy.

swarm in largest cluster over L

swarm,

ratio of

(k) inhomogeneous

After a fast increase for generations t < 15, a saturation in the best fitness is seen. The median of the last generation (t = 29) is 3.33 which means that the prediction network is predicting in average 83.3% of the sensor values correctly. In Fig. 4b a comparison of the last generation's best individual fitness is given for all settings of the ring length L(reported before, Hamann (2014)). The highest median best fitness is reached for L = 5 (3.68) and for L = 50(3.64) whereas the lowest median best fitness is obtained for L=25 (3.33). With increasing ring length (i.e., decreasing swarm density) the medians decrease (L < 30) and start to increase again for L > 25. From this result we infer that the prediction task is more difficult for medium density (10 $\leq L \leq$ 35) without excluding the possibility that better results could be achieved by investing more resources to the optimization process. For very high densities prediction seems easy because most of the time all sensors detect neighbors and hence the perception changes only infrequently. For very low densities prediction seems also easy because most of the time all sensors detect no neighbors. For medium densities, however, the detection of neighbors might change often hence always predicting the detection of neighbors or always predicting that no neighbors will be detected is insufficient.

Covered distance

In the following we investigate the evolved behaviors. This is done along the above mentioned two dimensions of basic collective behaviors, namely covered distance and distance between agents. The covered distance is measured over a period of time $\tau = \frac{L}{2v}$ at the end of the evaluation. It is the sum of all agents' covered distances. It is normalized by the length of the time period and the swarm size N. We get

$$D = \frac{1}{N\tau} \sum_{i \in \{0,1,\dots,N-1\}} |x_i(T-\tau) - x_i(T)|, \quad (2)$$

whereas $x_i(t)$ is the position of agent i at time t. The covered distance has a maximum of $D_{\rm max}=0.1=v$. In Fig. 4c the results averaged over the population of the last generation of the 200 independent evolutionary runs for varied ring length are given (note logarithmic scale). With increasing ring length (i.e., decreasing swarm density) the median covered distance increases (L<25) and starts to decrease again for L>20. Hence, it is very likely to see a static swarm for ring length L=50 and rather unlikely for L=20.

Largest cluster

As an indicator for the distances between agents we determine the largest cluster. For each agent we determine the total number of agents within a distance of s_1 (i.e., we count all agents that are in any of the four intervals of sensors L1, L0, R0, R1). The maximum over these numbers divided by swarm size N determines the size C of the largest cluster.

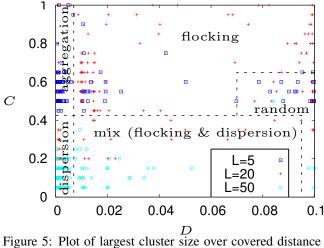


Figure 5: Plot of largest cluster size over covered distance for $L \in \{5, 20, 50\}$.

This measure is more meaningful than just calculating the sum over all agent-agent distances because it allows, for example, to distinguish situations when the swarm is separated in two flocks from situations with uniform distributions. The results averaged over the population of the last generation of the 200 independent evolutionary runs are given in Fig. 4d. For L=5 the median of the largest cluster size is C=0.55(11 agents) and hence relatively big as expected because an agent covers $2s_1$ with its sensors which is 40% of the ring for L=5. With increasing ring length the median of the largest cluster size increases (L < 20) and starts to decrease again for L > 15 until it reaches the low value of C = 0.15(3 agents) for L = 50. Hence, it is very likely to see an aggregated swarm, for example, for ring length L=15 and rather unlikely for L=50. To make a statement concerning the expected frequency of basic collective behaviors, both indicators (covered distance and largest cluster size) have to be considered in combination. For example, flocking is likely to occur for L=15 because fast and well aggregated swarms are frequent for that setting. However, from these results one cannot tell whether both features coexist. In Fig. 5 the largest cluster size is plotted over covered distance for $L \in \{5, 20, 50\}$ and hence allowing to classify the behaviors at least roughly. Aggregation and random behavior is mostly observed for L=5. Dispersion is mostly observed for L=50. Flocking is mostly observed for L=20, whereas it is possible to distinguish between slow moving flocks and fast moving flocks (cf. Fig. 2, flocks can move with different velocities). The behaviors seen in the region 0.07 < D < 0.095, C < 0.42 are inconclusive as they are mixes between flocking and dispersion (i.e., a fraction of the swarm moves as flock and another fraction is dispersed).

Entropy of sensor input

The difficulty of predicting future sensor inputs and minimizing surprise is conceptually connected to information theory by the concept of Shannon entropy. We are interested in the Shannon entropy of the sensor inputs. The entropy is measured for each agent over the whole period of the simulation. For each sensor $X \in \{L1, L0, R0, R1\}$ we measure the probability p_X that the sensor gives a 1 as output and calculate the entropy

$$H_X = -p_X \log_2(p_X) - (1 - p_X) \log_2(1 - p_X). \tag{3}$$

The overall entropy for the considered agent is the average over all sensors $H=\frac{1}{4}(H_{\rm L1}+H_{\rm L0}+H_{\rm R0}+H_{\rm R1}).$ Finally, we average over all individual entropies H of the swarm. The results averaged over the population of the last generation of the 200 independent evolutionary runs are given in Fig. 4e. Note that high entropy is measured for $L\in\{10,15,20,25,30\}$ which corresponds to relatively low fitness for the same ring lengths as seen in Fig. 4b. The inverse accordance of entropy and fitness indicates that diverse sensor input is inherent for intermediate swarm densities.

Control experiments - random genomes

A series of control experiments was done to test whether the results reported above are different from randomly generated behaviors and whether especially the random changes to the action networks by mutations are effective and relevant. A set of 200 random genomes was generated for each ring length $L \in \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$ and a swarm controlled by the respective action networks was simulated for each of them. The performance of these random prediction networks is low for all ring lengths. The median fitness is about 2.0, that is, in average the prediction is correct for two out of the four sensors. This would be expected for randomly generated predictions and equally distributed sensor values which is, however, not necessarily the case as it depends on swarm density and the actual agent behaviors. The entropy of the sensor input for these random behaviors is shown in Fig. 4f. For all ring lengths the entropy of random behaviors is significantly higher than for the entropy of evolved behaviors as shown in Fig. 4e. We also note that the difference between Figs. 4e and f is mostly a mere displacement by $\Delta H \approx 0.4$, that is, the overall shape of the curve is preserved. Hence, the entropy also depends inherently on the swarm density. Given that a randomly generated population is also the starting point of the evolutionary approach, it is justified to say that the evolution of prediction machines minimizes the entropy of the sensor input, too.

The covered distance D is shown in Fig. 4g. It is also invariant to the ring length (no significant differences) and it is in average shorter compared to the evolved behaviors. This is expected because by random control the swarm is unlikely to align and form a flock.

The size of the largest clusters C is shown in Fig. 4h. Large clusters are observed for L=5 as expected because an agent covers already 40% of the ring with its sensors. Still, the system cannot be understood as a mere point process because

the agents might react to each other based on their random controllers. For an assumed uniform distribution of agents we would expect largest clusters of size C=0.4 (8 agents), however, the median is 0.55 (11 agents). With increasing ring length L the size of the largest clusters decreases fast as expected. For ring lengths $L \in \{10, 15, 20, 25\}$ the largest clusters are significantly different and the evolved behaviors discussed above achieve bigger clusters, that is, better organized swarm behaviors. Interestingly, in the case of random behaviors there are a few extreme outliers even for large ring lengths up to L=50 reaching up to C=1 while the evolved behaviors achieve only $C\leq 0.5$ for L>40.

Control experiments - inhomogeneous swarm

Another series of experiments was done to test whether collective behaviors can also be evolved in an inhomogeneous swarm. Before all agents of a particular simulation run (simulated agents on ring) were clones because they shared the same genetic material. Now each agent of a simulation run has its own genome which most likely differs from all other genomes in the swarm. Hence, we merge the two populations that we had before, that is, the population of the evolutionary algorithm is identical to the population in the simulation runs. An advantage of this approach is that the need for computational resources is decreased significantly. Before we were doing 10 repetitions of simulation runs for each genome which sums to 500 simulation runs per generation. Now we do only 10 repetitions of simulation runs per generation because we evaluate all genomes within the same simulation run. However, this approach is ineffective as seen in Fig. 4i. While for ring length L=5 the approach is competitive to the homogeneous approach (cf. Fig. 4b, note different scales), it obtains poor fitness for bigger ring lengths. Also the results for covered distance (Fig. 4j) and size of the largest cluster (Fig. 4k) indicate the poor performance as they are similar to the results of the random genomes (Figs. 4g and h). The evolved inhomogeneous swarms always show random behavior (data not shown).

Discussion and Conclusion

Following the idea of the free-energy principle (Friston, 2010; Friston et al., 2006), that is, minimizing surprise (prediction errors), we have shown that a variety of basic collective behaviors can be evolved by selecting for the ability of making good predictions only. Without generating any direct selective pressure on the actual evolved behavior these typical swarm behaviors emerge depending on the swarm density. Our findings are supported by control experiments. The comparison to random behaviors as generated by random genomes are significantly different. The comparison to evolving behaviors in an inhomogeneous swarm shows that homogeneity in the genomes and hence in the behaviors of a swarm is crucial. However, we do not rule out the possibility to evolve interesting collective behaviors in an inhomo-

geneous swarm if the inhomogeneity is limited (e.g., initializing to a homogeneous population, minimizing the effect of mutations, greedy selection).

The comparison of this approach to the concept of novelty search (Lehman and Stanley, 2008; Mouret and Doncieux, 2009) is interesting. Novelty search generates selective pressure towards unpopulated regions of behavior space, that is, it tries to trigger the evolution of novel behaviors that were not observed before. Our approach of minimizing prediction error seems to implement the opposite, that is, it seems to reward repetitive and hence easy to predict behaviors (on the time scale of a genome's evaluation while novelty search operates on the time scale of generations). Still, the approach is effective and generates a variety of different behaviors. In contrast to novelty search no measure of behavioral distance needs to be defined. The application of this approach to goal-oriented evolution of collective behaviors requires additional research to determine how to guide the evolution. The similarity between the trajectory plots of the evolved behaviors especially for swarms showing flocking (see Fig. 2d) and space-time diagrams of cellular automata (e.g., see Crutchfield and Mitchell (1995, Fig. 2)) is probably more than mere coincidence. Also the concept of particles in cellular automata seems comparable to the concept of flocks here (see Crutchfield and Mitchell (1995, Table 2)). The reported swarm model is spatially 'almost discrete' (agents are displaced by a constant distance) and the agents move in continuous space only due to the continuous initialization of positions and the additive noise. Furthermore, the sensors also discretize an agent's neighborhood spatially (see Fig. 1a), hence the action network might implement simple update rules as seen in cellular automata.

There is also a low similarity to cellular automata in the context of computation 'at the edge of chaos' as discussed in (Langton, 1990; Crutchfield and Young, 1990). The results concerning the best fitness, the covered distance, and the size of the largest cluster (see Fig. 4b-d) indicate that for ring lengths of $L \in [10, 25]$ the evolutionary optimization problem is possibly more difficult and the required behaviors are possibly more complex. For high (L < 10) and for low swarm densities (L > 30) the sensory input is rather monotonous (see results of entropy measurements, Fig. 4e, a majority of sensors give frequently a '1' for high densities or a majority of sensors give frequently a '0' for low densities). For medium densities the sensory input is more likely to change over time and consequently more complex behaviors seem to emerge (e.g., flocking with flocks having different velocities depending on the flock size).

From the perspective of defining optimal conditions for collective behavior, the existence of an optimal density for good swarm performance is known (Hamann, 2013, 2012). The universality of these findings seems to be confirmed by the results concerning the best fitness, the covered distance, and the size of the largest cluster (see Fig. 4b-d).

Future work includes the investigation of how this approach can be utilized also as an approach to open-ended evolution to evolve desired behaviors similar to the concept of novelty search. Another subject will be to investigate whether minimizing prediction error triggers the evolution of positive and negative feedback loops within swarms.

References

- Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). Swarm Intelligence: From Natural to Artificial Systems. Oxford Univ. Press, New York, NY.
- Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2013). Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41.
- Buhl, J., Sumpter, D. J. T., Couzin, I. D., Hale, J. J., Despland, E., Miller, E. R., and Simpson, S. J. (2006). From disorder to order in marching locusts. *Science*, 312(5778):1402–1406.
- Capdepuy, P., Polani, D., and Nehaniv, C. L. (2007). Maximization of potential information flow as a universal utility for collective behaviour. In *Proceedings of the 2007 IEEE Symposium* on Artificial Life, pages 207–213.
- Crutchfield, J. P. and Mitchell, M. (1995). The evolution of emergent computation. *Proc. Natl. Acad. Sci. USA*, 92:10742–10746.
- Crutchfield, J. P. and Young, K. (1990). Computation at the onset of chaos. In Zurek, W., editor, *Complexity, Entropy, and Physics of Information*. Addison-Wesley, Reading, MA.
- Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. (1995). The Helmholtz machine. *Neural Computation*, 7(5):889–904.
- Dorigo, M. and Şahin, E. (2004). Guest editorial: Swarm robotics. *Autonomous Robots*, 17(2-3):111–113.
- Friston, K. (2010). The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2):127–138.
- Friston, K., Kilner, J., and Harrison, L. (2006). A free energy principle for the brain. *J. of Physiology Paris*, 100(1):70–87.
- Gomes, J., Urbano, P., and Christensen, A. L. (2013). Evolution of swarm robotics systems with novelty search. *Swarm Intelli*gence, 7(2-3):115–144.
- Hamann, H. (2010). Space-Time Continuous Models of Swarm Robotics Systems: Supporting Global-to-Local Programming. Springer, Berlin, Germany.
- Hamann, H. (2012). Towards swarm calculus: Universal properties of swarm performance and collective decisions. In Dorigo, M., et al., editors, Swarm Intelligence: 8th International Conference, ANTS 2012, LNCS 7461, pages 168–179, Springer.
- Hamann, H. (2013). Towards swarm calculus: Urn models of collective decisions and universal properties of swarm performance. Swarm Intelligence, 7(2-3):145–172.
- Hamann, H. (2014). Evolving prediction machines: Collective behaviors based on minimal surprisal. In *Int. Conf. on Genetic and Evolutionary Computation (GECCO 2014)*. ACM. [extended abstract], in press.

- Hammer, M. and Menzel, R. (1995). Learning and memory in the honeybee. *The Journal of Neuroscience*, 15(3):1617–1630.
- Langton, C. G. (1990). Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D*, 42(1-3):12–37.
- Lehman, J. and Stanley, K. O. (2008). Exploiting open-endedness to solve problems through the search for novelty. In Bullock, S., Noble, J., Watson, R., and Bedau, M. A., editors, *Artificial Life XI: Proc. of the 11th Int. Conf. on the Simulation and Synthesis of Living Systems*, pages 329–336. MIT Press.
- Martinoli, A., Easton, K., and Agassounon, W. (2004). Modeling swarm robotic systems: A case study in collaborative distributed manipulation. *Int. Journal of Robotics Research*, 23(4):415–436.
- Mouret, J.-B. and Doncieux, S. (2009). Using behavioral exploration objectives to solve deceptive problems in neuro-evolution. In *Proc. of the 11th Conf. on Genetic and Evolutionary Computation (GECCO'09)*, pages 627–634. ACM.
- Nelson, A. L., Barlow, G. J., and Doitsidis, L. (2009). Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57:345–370.
- Nolfi, S. and Floreano, D. (2000). Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. MIT Press.
- Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2):265–286.
- Schmickl, T. and Crailsheim, K. (2006). Bubbleworld.Evo: Artificial evolution of behavioral decisions in a simulated predator-prey ecosystem. In *From Animals to Animats 9*, volume 4095 of *LNCS*, pages 594–605. Springer.
- Trianni, V. (2008). Evolutionary Swarm Robotics: Evolving Self-Organising Behaviours in Groups of Autonomous Robots, Springer.
- Trianni, V., Groß, R., Labella, T. H., Şahin, E., and Dorigo, M. (2003). Evolving aggregation behaviors in a swarm of robots.
 In Banzhaf, W., et al., editors, Advances in Artificial Life (ECAL 2003), volume 2801 of Lecture Notes in Artificial Intelligence, pages 865–874. Springer.
- Trianni, V., Labella, T. H., and Dorigo, M. (2004). Evolution of direct communication for a swarm-bot performing hole avoidance. In Dorigo, M., et al., editors, *Ant Colony Optimization and Swarm Intelligence (ANTS 2004)*, volume 3172 of *LNCS*, pages 130–141. Springer.
- Trianni, V., Tuci, E., Passino, K. M., and Marshall, J. A. (2010). Swarm cognition: an interdisciplinary approach to the study of self-organising biological collectives. *Swarm Intelligence*, 5(1):3–18.
- von Helmholtz, H. (1867). *Handbuch der physiologischen Optik*. Ludwig Voss, Leipzig, Germany.
- Ward, C. R., Gobet, F., and Kendall, G. (2001). Evolving collective behavior in an artificial ecology. *Artificial Life*, 7(2):191–209.

An informational study of the evolution of codes in different population structures

Andrés C. Burgos¹ and Daniel Polani²

^{1,2}Adaptive Systems Research Group, University of Hertfordshire, Hatfield, UK ¹a.c.burgos@herts.ac.uk

Abstract

We consider the problem of the evolution of a code within a structured population of agents. The agents try to maximise their information about their environment by acquiring information from the outputs of other agents in the population. A naive use of information-theoretic methods would assume that every agent knows how to "interpret" the information offered by other agents. However, this assumes that one "knows" which other agents one observes, and thus which code they use. In our model, however, we wish to preclude that: it is not clear which other agents an agent is observing, and the resulting usable information is therefore influenced by the universality of the code used and by which agents an agent is "listening" to.

Introduction

If we consider organisms capable of processing information, then we can argue that they must be able to internally assign meaning to the symbols they perceive in a code-based manner (Görlich et al., 2011). For instance, bacteria perceives chemical molecules in their environment and interprets them in order to better estimate environmental conditions and (stochastically) decide their phenotype (Platt and Fuqua, 2010). Plants detect airborne signals released by other plants, being able to interpret them as attacks of pathogens or herbivores (Heil and Karban, 2010). Therefore, a correspondence between environmental conditions and chemical molecules must be established. It is in this way that Barbieri characterises codes, and he proposes three fundamental characteristics for them: they connect two independent worlds; they add meaning to information; and they are community rules (Barbieri, 2003).

Codes connect two independent worlds by establishing a correspondence or mapping between them. These worlds are independent and thus there are no material constraints for establishing arbitrary mappings. The meaning of information comes exclusively from the mapping: symbols by themselves are meaningless. Finally, the third property requires that the correspondence between the two worlds constitutes an integrated system.

For instance, human languages establishes a correspondence between words and objects (Barbieri, 2003); in bacteria it is between chemical molecules and environmental conditions (Waters and Bassler, 2005). Words (or chemical molecules) by themselves do not have any meaning, they are chosen arbitrarily and any individual of a population can define its own set together with its mapping. However, populations of individuals sharing the same code are ubiquitous in nature. How is it that codes come to be shared by many individuals when, as stated, they are completely arbitrary? This question is what we are investigating in the present paper.

For this work, we assume a simple scenario where organisms seek to maximise their long-term growth rate by following a bet-hedging strategy (Seger and Brockmann, 1987). We know that maximising their information about the environment achieves this (Shannon, 1948; Kelly, 1956; Donaldson-Matasci et al., 2010). Then, individuals obtaining side environmental information from other individuals will have an advantage over those that do not, since they would be able to better predict the future environmental conditions. However, for individuals to be able to communicate with each other, they must be able to translate symbols into environmental conditions, where the

output of these symbols results from an individual's code. The code of an individual is a stochastic mapping from its sensors states to a set of outputs.

For this study, we consider outputs of individuals (or agents) as conventional signs. In semiotics, the science of all processes in which signs are originated, stored, communicated, and being effective (Görlich et al., 2011), two types of signs are traditionally recognised: *conventional signs* and *natural signs* (Deely, 2006). In conventional signs there is no physical constraint on the possible mappings, they are established by conventions. On the other hand, in natural signs, there is always a physical link between the signifier and signified, such as smoke as a sign of fire, odours as signs of food, etc. (Barbieri, 2008).

We are not interested in the particular detailed mechanisms by which an agent implements its code, nor how the agent decodes the outputs of other agents. Instead, we focus on the theoretical limits on the amount of environmental information an agent can possibly acquire resulting from different scenarios of population structure and codes distribution.

The natural framework to analyse such quantities is information theory (Shannon, 1948). However, it does not take semantic aspects into account, it only deals with frequencies of symbols instead of what they symbolise. Codes, on the other hand, add meaning to information, which makes the integration of sciences such as semiotics with information theory non-trivial (Favareau, 2007; Battail, 2009). In the following section, we present an information-theoretic model which incorporates the necessity of conventions by dropping from the model the usual implicit assumption of knowing the identity of the communicating units.

Model

To introduce the model in a progressive manner, let us first consider three agents, θ_1 , θ_2 and θ_3 . Each of these agents depend on the same environmental conditions for survival, which are modelled by a random variable μ . Agents acquire information about the environment through their sensors, which are modelled by random variables Y_{θ_1} , Y_{θ_2} and Y_{θ_3} , all three conditioned on

 μ , for agents θ_1 , θ_2 and θ_3 , respectively. We assume each agent acquires the same amount and aspects of environmental information from μ , *i.e.* $p(Y_{\theta_1}|\mu) = p(Y_{\theta_2}|\mu) = p(Y_{\theta_3}|\mu)$. Let us further assume that the information each agent acquires about the environment does not eliminate its uncertainty, *i.e.* $H(\mu|Y_{\theta_i}) > 0$ for $1 \le i \le 3$. The code of an agent is a stochastic mapping from its sensor states into a set of outputs, and is represented by the conditional probabilities $p(X_{\theta_1}|Y_{\theta_1})$, $p(X_{\theta_2}|Y_{\theta_2})$ and $p(X_{\theta_3}|Y_{\theta_3})$ for agents θ_1 , θ_2 and θ_3 , respectively (see Fig. 1).

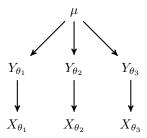


Figure 1: Bayesian network representing the relantionship between the sensor and output variables of three agents.

Let us assume that agent θ_1 perceives only the outputs of agents θ_2 and θ_3 . One possible way of computing the information about the environment agent θ_1 has is to consider the mutual information between μ and the joint distribution of the sensor of θ_1 and the outputs of θ_2 and θ_3 : $I(\mu; Y_{\theta_1}, X_{\theta_2}, X_{\theta_3})$. However, by writing down this quantity, we are implicitly assuming that agent θ_1 "knows" which output corresponds to θ_2 and which output corresponds to θ_3 . Therefore, in this consideration, an agent can theoretically do the translations of the outputs according to some internal model of other agents and infer the mentioned amount of information about its environment.

Indistinguishable sources

For this study, on the contrary, we consider an agent observing other agents' messages, but under the assumption that the originator of a message cannot be identified. In this way, the total amount of information an agent can infer from the outputs of other agents will depend on to which extent it either can identify who the other agents are or can rely on them using a coding scheme that does not depend too much on their particular identity. For instance, if agents θ_2 and θ_3 both

agree on the output for each of the environmental conditions, then agent θ_1 should be able to infer more environmental information than if they disagree on the output for each of the environmental conditions, given that agent θ_1 does not know which of the agents it is observing.

To model this idea, let us assume a random variable Θ' denoting the selected agent, which depends on the same environmental conditions for survival, which are modelled, as above, by a random variable μ . Agents acquire information about the environment through their sensors, which are modelled by a random variable $Y_{\Theta'}$ conditioned on the index variable denoting the agent under consideration, Θ' , and μ . The amount of acquired sensory information of a specific agent θ' about μ is given by $I(\mu;Y_{\theta'})$. As above, the code of an agent is a stochastic mapping from its sensor states into a set of outputs, and is represented by the conditional probability $p(X_{\theta'}|Y_{\theta'})$ for an agent θ' (see Fig. 2).

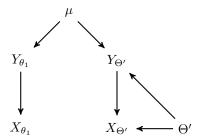


Figure 2: Bayesian network representing the relationships as described above (see text).

However, now we want to model the fact that we do not know which agent is observed. In the case with maximum uncertainty, Θ is uniformly distributed, and then this parametrisation of the codes considers the outputs of all agents in Θ' altogether, such that if we are not observing Θ' , we cannot identify whose agent's output we are observing. In Eq. 3 and Eq. 4 we show two examples of codes for agents θ_2 and θ_3 , while their sensor states are define by the Eq. 2 (Eq. 1 defines the sensors states of agent θ_1). We compute how much information about the environment there is when the outputs of both agents (θ_2 and θ_3) are considered together by agent θ_1 .

$$Pr(Y_{\theta_1}|\mu) := \frac{\mu_1}{\mu_2} \begin{pmatrix} y_1 & y_2 \\ 0.99 & 0.01 \\ 0.01 & 0.99 \end{pmatrix} \tag{1}$$

$$Pr(Y_{\Theta'}|\mu,\Theta') := \begin{cases} \theta_2, & \mu_1 \\ \theta_2, & \mu_2 \\ \theta_3, & \mu_1 \\ \theta_3, & \mu_2 \end{cases} \begin{pmatrix} y_1 & y_2 \\ 0.99 & 0.01 \\ 0.99 & 0.01 \\ 0.01 & 0.99 \end{pmatrix}$$
(2)

$$Pr(X_{\Theta'}|Y_{\Theta'},\Theta') := \begin{array}{c} \theta_2, \ y_1 & 1 & 0\\ \theta_2, \ y_2 & 0 & 1\\ \theta_3, \ y_1 & 1 & 0\\ \theta_3, \ y_2 & 0 & 1 \end{array}$$
(3)

$$Pr(X_{\Theta'}|Y_{\Theta'},\Theta') := \begin{cases} \theta_2, \ y_1 & 1 & 0\\ \theta_2, \ y_2 & 0 & 1\\ \theta_3, \ y_1 & 0 & 1\\ \theta_3, \ y_2 & 1 & 0 \end{cases}$$
(4)

If we assume $p(\theta_2) = p(\theta_3) = 1/2$, and $p(\mu_1) =$ $p(\mu_2) = 1/2$, then if we consider the codes shown in Eq. 3, we have that $I(\mu; Y_{\theta_1}, X_{\Theta'}) = 0.97872$ bits, where Θ' consists of agents θ_2 and θ_3 . However, had θ_2 and θ_3 "opposite" codes as shown in Eq. 4, then $I(\mu; Y_{\theta_1}, X_{\Theta'}) = 0.9192$ bits, which is exactly $I(\mu; Y_{\theta_1})$, that is, $I(\mu; X_{\Theta'}|Y_{\theta_1}) = 0$ bits (agent θ_1 cannot acquire any side information from the outputs of agents θ_2 and θ_3). We should note here that the sensor states y_1 and y_2 of agents θ_2 and θ_3 in the conditional probability shown in Eq. 3 and 4 refer almost deterministically to the same environmental condition, and therefore the loss of side information is thus entirely due to the incompatible codes. The conditional probabilities of sensor states given the environmental conditions further defined throughout the paper are also assumed to be almost deterministically.

Population information

The model shown in Fig. 2 considers the environmental information of agent θ_1 , ignoring its own output X_{θ_1} . Nevertheless, agents ignoring their outputs is contrary to our assumption over the sources of the outputs. To incorporate this option in the model shown in Fig. 2, we could consider the state space of Θ' as the set $\{\theta_1, \theta_2, \theta_3\}$. Then, to express not only the environmental information of agent θ_1 , but the average environmental information of the whole population, we can parametrise the sensors of the agents by a random variable Θ (defined over the same state space, representing the same set of agents as Θ'), such that

 $p(Y_{\Theta}|\mu,\Theta) = p(Y_{\Theta'}|\mu,\Theta')$ (i.e., $Y_{\Theta'}$ is i.i.d. to Y_{Θ} , and vice versa).

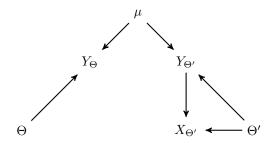


Figure 3: Bayesian network representing the sensor variables of a set of agents indexed by the random variable Θ , and the sensor and output variables of a copy of the set of agents indexed by Θ named Θ' .

In this way, the average environmental information of a population of the agents selected by Θ is given by $I(\mu; Y_{\Theta}, X_{\Theta'})$ (see Fig. 3). This measure can be consider as the objective function to maximise in our model. However, we would be making two important assumptions: first, this objective function assumes agents have access to the environmental conditions μ , which they indirectly do but only through their sensors; and second, every agent would perceive the output of every other agent, including itself. In this work, we instead simplify the model in that we propose agents following a behaviour such that it maximises the similarity of their outputs (via their codes) with those of which the agent perceives. A consequence of this behaviour is that the average information about μ is also maximised. In addition, we will introduce a potentially flexible "population structure", so that we can specify which agents interact with which.

Code similarity

First, we introduce a copy of the codes of the agents, such that when we instantiate the variables X_{Θ} and $X_{\Theta'}$, the probabilities are the same. The structure of the population is then given by $p(\Theta, \Theta') = p(\Theta)p(\Theta')$. However, the conditional independence of Θ and Θ' restricts significantly the diversity of the structures that can be represented. In order to model a general interaction structure between agents, we consider $p(\Theta, \Theta')$ not independent, as shown in the Bayesian network in Fig. 4, where we introduce a helper variable Ξ .

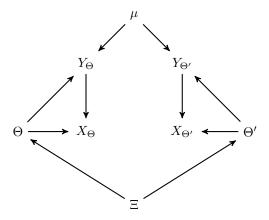


Figure 4: Bayesian network representing the relantionship of the variables in the model of code evolution. $Y_{\Theta'}$ is an *i.i.d.* copy of Y_{Θ} and $X_{\Theta'}$ is an *i.i.d.* copy of X_{Θ} . Θ' covers the same set of agents as Θ , but its probability distribution is not necessary the same.

We can interpret our objective function $I(X_\Theta; X_{\Theta'})$ as the average code similarity of a population of agents according to the population structure $p(\Theta, \Theta')$. For instance, if two agents cannot exchange outputs in a given structure, then there is no gain (in the value of the objective function) in adopting similar codes for these two agents.

If we consider our system as a process in time, then at each time-step two agents are chosen according to $p(\Theta, \Theta')$. Agent Θ reads the output of agent Θ' (generated via its code, which is *i.i.d* over time), and let us assume that it stores the pair $(Y_{\Theta}, X_{\Theta'})$, i.e. its current sensor state together with the perceived output. If this is repeated a large number of times, then the total amount of environmental information that can be inferred from the collected statistics by the population is bounded by $I(\mu; Y_{\Theta}, X_{\Theta'})$. This is the theoretical limit to which we refer in the introduction, and for this study we are not interested in how the inference is computed. However, we implicitly assume that agents decode the perceived outputs according to their codes.

Distance between two codes

In order to visualise the evolution of codes, we define the distance between the codes of two agents θ_i and θ_j as the square root of the *Jensen-Shannon divergence* (Wong and You, 1985; Lin, 1991) be-

tween them. This measure has the property that $0 \leq JSD(\theta_i,\theta_j) \leq 1$ when \log_2 is used, and the square root yields a metric. Let us note that this distance requires the sensor states Y to be named identically (for the corresponding states of μ) among agents in order to be meaningful. As we stated above, this is (closely) the case in all our experiments. This requirement over the sensor states discards the possibility of using other measures such as mutual information.

Methods

To illustrate the behaviour of our model, we consider three different scenarios, which are described in the Results section. The common parameters for the first two experiments are the following: the population consists of 25 agents (the small number was chosen to avoid high computational costs); the amount and quality of the acquired sensory information is the same for every agent, that is $p(Y_{\theta_i}|\mu) = p(Y_{\theta_j}|\mu)$ for every $i,j \in [1,25]$. For the last scenario, the only difference is that we consider only 15 agents.

The optimisation algorithm used in the following experiments is CMA-ES (Covariance Matrix Adaptation Evolution Strategy), which is a stochastic derivative-free method for non-linear optimisation problems (Hansen and Ostermeier, 2001). We utilised the implementation provided by the Shark library v3.0.0 (Igel et al., 2008) with its default parameters, which implements the CMA-ES algorithm described in (Hansen and Kern, 2004). The evolutionary algorithm used for optimisation does not intend to represent the actual evolution of the codes. Instead, we are interested in the solutions of this optimisation process, which are representative of the possible outcomes of evolution.

To visualise the evolution of the codes of the agents, we use the method of multidimensional scaling provided by R version 2.14.1 (2011-12-22).

Results

In this section, we analyse the outcome of three different scenarios where code similarity is maximised. While the outcomes are particular for one

simulation, they are illustrative of the richness that the model is able to capture, which is described for each scenario. The outcomes are typical solutions, and we cannot perform statistics over simulations since the many solutions are qualitatively different.

Well-mixed population

In the first scenario, each agent θ_i perceives the output of every other possible agent θ_j with the same probability, that is $p(\theta_i,\theta_j)=1/25^2$ for every $i,j\in[1,25]$. The maximum average code similarity is bounded by $I(Y_\Theta;Y_{\Theta'})=1.71908$ bits, which is achieved under two conditions: first, every code must be a one-to-one mapping; second, the code must be universal. This is indeed the outcome of the performed optimisation, as we show in Fig. 5: the optimised codes (blue points) converged into a universal code (the distance between any of them is zero). Each red point correspond to an initial code.

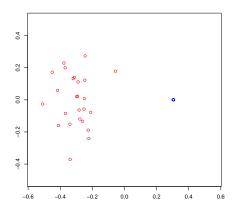


Figure 5: 2-dimensional plot of code distance: red points are codes at the beginning of the optimisation process; blue points are codes at the end of the optimisation process (where the distance between every pair of codes is zero).

The resulting code adopted by the population is a one-to-one mapping between sensor states and outputs, and any of the 24 possible one-to-one mappings is a global maximum (there are 4 sensor states and 4 possible outputs). However, it is still interesting to briefly analyse the possible paths towards a universal and optimal code. In Fig. 6, we show the distribution of the adopted codes by the agents of the population in a moment of the optimisation process where the average code similarity is $I(X_{\Theta}; X_{\Theta'}) = 1.18276$ bits. Here, the most popular code is the suboptimal code shown

in Fig. 6 (a). This results from the particular initialised codes, driving the agents temporarily towards a suboptimal code. However, once any of the many-to-one codes becomes (nearly) universally adopted, then any code's deviation improving code similarity will eventually drive the convention towards optimality. The fact that it does not need simultaneous changes in the code increases the likeliness of improving the code similarity.

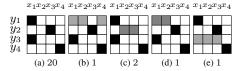


Figure 6: Representation of the codes p(x|y) by a heat-map using inverse grayscale. For each evolved code, we output the number of agents adopting it. This code distribution was achieved with 25 agents in a well-mixed population.

Spatially-structured population

In another set-up, we assume the agents are structured in a 5×5 grid, where $p(\theta,\theta')=1/105$ if θ and θ' are neighbours or when $\theta=\theta'$. After randomly initialising the codes, the performed optimisation plateaued on an average code similarity of $I(X_\Theta;X_{\Theta'})=1.13536$ bits. As in the former scenario, here the optimal solution is also a universal code with a one-to-one mapping. However, in this case, the result is not a universal code, as can be appreciated in Fig. 7. Spatially structure populations are sensitive to the initial codes and how codes are updated.

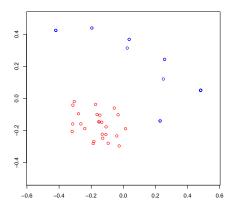


Figure 7: 2-dimensional plot of code distance: red points are codes at the beginning of the optimisation process; blue points are codes at the end of the optimisation process.

The resulting code distribution among the population is shown in Fig. 8, with 8 different codes in the population. Different from the well-mixed population structure, in a spatially structured population the pressure to agree on a code occurs only between neighbours. A consequence of this is that many local conventions are established within neighbourhoods, and once this situation is reached, to improve the total code similarity, some simultaneous changes to the agent's codes would be needed. For instance, the code shown in Fig. 8 (e) could increase the average similarity of the population if $p(x_2|y_1) = 1$, as it is in the rest of the codes. However, for this to happen (in this particular case), at least two agents need to change their code simultaneously (otherwise the average similarity decreases), which makes the deviation from the resulting code distribution unlikely.

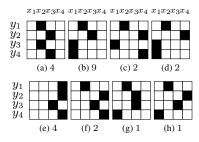


Figure 8: Representation of the codes p(x|y) by a heat-map using inverse grayscale. For each evolved code, we output the number of agents adopting it. This code distribution was achieved with 25 agents in a grid structure.

Free structure

For our last scenario, we let the structure evolve with the codes without any constraint. In this case, the resulting average code similarity is nearly optimal, but the code is not necessarily universal. This is because, when the structure is not fixed, agents form roughly disconnected clusters of related codes. In this process, the interaction probability of agents with unrelated codes will vanish, decreasing the entropy of the population structure $H(\Xi)$ (see Fig. 9). However, once the clusters are formed, if it is not a single isolated agent (such that nobody perceives its output), then each cluster conform a universal code within itself. In the latter case, the entropy of the structure can increase if the agents within a cluster perceive the outputs of all other agents also within their clusters (periods where $H(\Xi)$ increases on in Fig. 9).

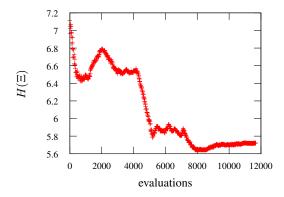


Figure 9: Evolution of the entropy of the population structure.

This is exemplified by the code distribution and population structure we obtained (see Fig. 10).

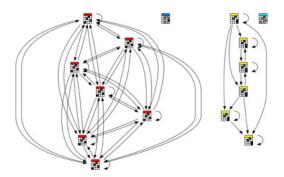


Figure 10: Each node in the graph corresponds to the code of an agent. There is a weighted edge between agent θ_i and θ_j if $p(\theta_i,\theta_j)>0$ (which is the weight). The temperature colours on top of the nodes indicate the amount of environmental information they would contribute to any agent perceiving only that agents output.

Discussion

We considered three different scenarios of code evolution: in the first one, all agents perceived the outputs of all other agents, including itself. We argued that two main stages of evolution can be recognised: in the first stage, a universal code is established, which can be optimal or not. If it is not optimal, then a second stage will achieve optimality. The same result was obtained in (Vetsigian et al., 2006), in a model of the evolution of the genetic code (represented as a probabilistic mapping between codons and amino acids), although universality and optimality were simultaneously achieved.

In the mentioned work, which developed further the ideas of (Woese, 2002, 2004), the authors argue that the universality of the genetic code is a consequence of early communal evolution, mediated by horizontal gene transfer (HGT) between primitive cells. In this evolutionary process, they argue, larger communities will have access (through the exchange of genetic material) to more innovations, leading to faster evolution than smaller ones. Then, "it is not better genetic codes that give an advantage but more common ones" (Vetsigian et al., 2006). Although their model does not explicitly show this property, it is captured in our model. We show that a more common, but not optimal code is widely adopted within a population (see Fig. 6). However, in our model, a code imposes itself as universal not because it provides access to more innovations (in our model there is no "code exchange", only the outputs are shared), but because the population structure forces the adoption of the most popular code. After this stage, further changes in the code of the agents eventually lead to optimality.

In another related work, (Oudeyer, 2005) explored the origins of language in a scenario consisting of artificial agents with a coupled perception and production of speech sounds. Although this work is focused on plausible mechanisms for the origin of language, it assumes the same similarity principle as we do (hearing a vocalisation increases the probability of producing similar vocalisations), arriving to the same outcome (a universal language, or code).

Our second scenario, where the structure of the population is a grid, showed how establishing local conventions in early stages of evolution constrains the outcome of the code distribution, since to reconcile different conventions, several simultaneous changes are needed. On the other hand, in our third scenario, where we let the structure of the population change simultaneously with the codes themselves, such situations are avoided by "disconnecting" clusters with dissimilar conventions. This property enhances evolution, and can potentially lead to the adoption of several different conventions within an (increasingly fragmenting, or "speciating") population.

The evolution and establishment of conventional codes as defined by Barbieri could be interpreted, in the widest sense, as a form of cultural evolution. While communication between individuals of a population opens up the possibility of "signal cheaters", our model does not allow such behaviour, since the code producing the outputs functions, implicitly, as the interpreter of the perceived signals.

Conclusion

In the proposed model, we introduced a key assumption which allowed us to evolve, for some structures, universal and optimal codes. This assumption states that an agent cannot distinguish the sources of the outputs it perceives from other agents. Following from this, a universal code will necessary introduce semantics by relating symbols to environmental conditions (via the internal states of the agent). Our model proposes an information-theoretic way of measuring the similarity within a population of codes.

In this work, we proposed, as an evolutionary principle, that agents try to maximise their side information about the environment indirectly by maximising their mutual code similarity. This behaviour produces several interesting outcomes in the code distribution of a structured population. Depending on the population structure, it captures the evolution of a universal and optimal code (well-mixed population structure), while also the evolution of different codes organised in clusters (in a freely evolving structure), which allows the establishment of optimal as well as suboptimal conventions.

References

- Barbieri, M. (2003). The organic codes-An introduction to semantic biology. *Genetics and Molecular Biology*.
- Barbieri, M. (2008). Biosemiotics: a new understanding of life. *Die Naturwissenschaften*, 95(7):577–99.
- Battail, G. (2009). Applying Semiotics and Information Theory to Biology: A Critical Comparison. *Biosemiotics*, 2(3):303–320.
- Deely, J. (2006). On 'semiotics' as naming the doctrine of signs. *Semiotica*.
- Donaldson-Matasci, M. C., Bergstrom, C. T., and Lachmann, M. (2010). The fitness value of information. *Oikos*, 119(2):219–230.

- Favareau, D. (2007). The evolutionary history of biosemiotics. *Introduction to biosemiotics*, pages 1–67.
- Görlich, D., Artmann, S., and Dittrich, P. (2011). Cells as semantic systems. *Biochimica et biophysica acta*, 1810(10):914–23.
- Hansen, N. and Kern, S. (2004). Evaluating the CMA evolution strategy on multimodal test functions. Parallel Problem Solving from Nature-PPSN VIII.
- Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–95.
- Heil, M. and Karban, R. (2010). Explaining evolution of plant communication by airborne signals. *Trends in ecology & evolution*, 25(3):137–44.
- Igel, C., Heidrich-meisner, V., and Glasmachers, T. (2008). Shark. *Journal of Machine Learning Research*, 9:993–996.
- Kelly, J. (1956). A new interpretation of information rate. *IEEE Transactions on Information Theory*, 2(3):185–189.
- Lin, J. (1991). Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151.
- Oudeyer, P. (2005). The self-organization of speech sounds. *Journal of Theoretical Biology*.
- Platt, T. G. and Fuqua, C. (2010). What's in a name? The semantics of quorum sensing. *Trends in microbiology*, 18(9):383–7.
- Seger, J. and Brockmann, H. J. (1987). What is bethedging? Oxford surveys in evolutionary biology.
- Shannon, C. (1948). A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423.
- Vetsigian, K., Woese, C. R., and Goldenfeld, N. (2006). Collective evolution and the genetic code. Proceedings of the National Academy of Sciences of the United States of America, 103(28):10696–10701.
- Waters, C. M. and Bassler, B. L. (2005). Quorum sensing: cell-to-cell communication in bacteria. *Annual review of cell and developmental biology*, 21:319–46.
- Woese, C. R. (2002). On the evolution of cells. Proceedings of the National Academy of Sciences of the United States of America, 99(13):8742–7.
- Woese, C. R. (2004). A new biology for a new century. Microbiology and Molecular Biology Reviews, 68(2):173–186.
- Wong, a. K. and You, M. (1985). Entropy and distance of random graphs with application to structural pattern recognition. *IEEE transactions on pattern analysis and machine intelligence*, 7(5):599–609.

In between Schelling and Maynard-Smith

Hugues Bersini

IRIDIA/ULB

50, av. Franklin Roosevelt 1050 Bruxelles - Belgium bersini@ulb.ac.be

Abstract

In the history of agent-based/Alife simulations helping to better understand some unexpected social consequences emerging out of interactive individual behaviors, two models have become quite celebrated: the Schelling's segregation model and the Maynard-Smith's emergence of cooperation in an albeit very defective world. In this paper, we explore a simulation at the crossroad of these two models. We show how in a bi-color society in which individual can move according to their neighborhood "colored" composition (reminiscent of the original Schelling's model), one further way to favor cooperation is to encourage "communitarian" behavior i.e. allowing individual to move, compose clusters of similar color and evolve a cooperative behavior restricted to partners of the same color. In brief, our results not so surprisingly tend to show that communitarian cooperation and segregation evolve hand in hand.

Introduction

Despite or on account of its simplicity, the Schelling's segregation model (Schelling, 1978) has become quite a classic of the sociological literature. It surprisingly shows how even a weakly communitarian attitude can drive to a very segregated society. The simulation on a 2-D board goes as follows. Each agent has one out of two colors. It is located on one site and the board contains a small fraction of empty sites. The neighborhood is a Moore's one. At each time step, an agent moves (and occupies a free location picked randomly) if more than two of its neighbors are of different color. The simulation shows (like in figure 1) how clusters of agents with same color rapidly percolate through the board despite a somewhat "tolerant" moving condition.

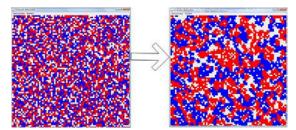


Figure 1: Schelling's segregation simulation: the starting and the final boards. Clusters of same color agents emerge despite weak communitarian rules of movement.

In another quite famous simulation (initially due to Maynard-Smith then further elaborated by Novak and many others – (Maynard-Smith, 1982; Novak, 2006; Novak et al., 1994)) based on the prisoner dilemma game, two types of interacting agents compete for occupying the full board: cooperators and defectors. The agents don't move but switch their type in time in order to imitate their most successful neighbors. At each time step, first all agents play a prisoner dilemma game with all their neighbors and cumulate their fitness according to the rule of the game indicated below.

Agent1/Agent2	Cooperator	Defector
Cooperator	1,1	0,1+x
Defector	1+x,0	0,0

As indicated in this gain matrix (and similar to the values tested by (Novak, 1994; 2006)), the pressure of "defection" as compared to the cooperative attitude depends on the value of "x". Then, at each time step also, an agent i selected randomly changes its "type" and imitates its neighbor j according to the following probability:

$$P_{i->j} = (f_i - f_i)/(F_{max} - F_{min})$$
 $if f_i > f_i$

Where f_j is the cumulated fitness of neighbor j, f_i the cumulated fitness of the agent to update and the denominator is based on the greatest and the lowest cumulated fitness of all agents. In agreement with results widely discussed in (Novak, 2006), whereas not so surprisingly defection invades the world for x > 0.25 and cooperation for x < 0.05, a much more interesting phenomenon (like shown in figure 2) unfolds for intermediary defection such as for x=0.15.

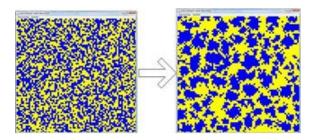


Figure 2: Maynard-Smith and Novak's evolutionary game simulation: cluster of cooperators can survive while surrounded by defecting agents.

In this simulation, clusters of cooperators can still prevail while being besieged by defectors. The way cooperation can emerge in a world subject to a very defective pressure has attracted a huge attention these last years. As Novak synthetizes in (Novak, 2006), common to the many possible alternatives such as "spatial clustering" (the one studied in this paper), "tit-for-tat", "social scale-free network", "group selection", always underlie the very intuitive idea that any possibility to make cooperators restrict their interaction with other similar cooperators favor their resistance to defection. For instance, both "tit-for-tat" and "spatial clustering" condemn defectors to reciprocate, the first in time and the second in space, and suppress them out. Reciprocation instead is much more advantageous for cooperators.

In this paper, we explore a further interesting alternative road, somewhere in between Schelling and Maynard-Smith, in which the simulation allows agent of two colors to successively move and play the interactive game. Our experimental results show how agents, simultaneously "movers" and "cooperators", can drive the world to both "segregate" and privilege a restricted form of "cooperation" with same color agents. The next chapter will describe the simulation in all details and the third one will present the experimental outcomes of this simulation.

The simulation

The following UML class and sequence diagrams are the best way to expose the different parts and the behavior of our simulator. In (Bersini, 2012), the same two UML diagrams (for a first and easy introduction to UML see (Fowler, 2003)) are exploited to respectively explain the Schelling and the Maynard-Smith/Novak simulations. Let's first describe the various classes of the code as presented in the first diagram. The world is a toroidal two dimensional grid composed of sites. The neighborhood is the Moore's one. Each agent is located on one site and some sites are left unoccupied. Each agent has one out of two colors. The agents can behave according to two main behavioral super classes: "Movement" and "Interaction". The "Movement" cares for the Schelling's part of the model while the "Interaction" cares for the Maynard-Smith's part of it. Let's first deal with the easy part: the movement behavior. Three sub-behaviors are possible.

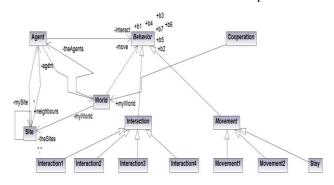


Figure 3.1 Class diagram of the model

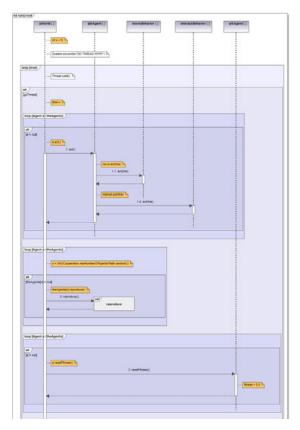


Figure 3.2: Sequence diagram of the simulation

- Movement1(stay): Paradoxically, this is the "immobile" behavior. The agent chooses not to move.
- Movement2: The agent moves to a random free site if its neighborhood is composed of a majority of agents of its color (this is the "anti-communitarian" attitude).
- 3. *Movement3*: The agent moves to a random free site if its neighborhood is composed of a minority of agent of its color (this is the communitarian attitude).

Notice that in the case the neighborhood is composed of an equal mix of colors, the agents can randomly decide to move or not. The two last behaviors (the real movements) receive a fitness penalty so that the "immobile" strategy is selectively always favored.

Now let's deal with the more subtle "interaction" part. Four sub-behaviors turn out to be possible, the most original being the last two.

- 1. Interaction1: Cooperate with all agents.
- 2. Interaction2: Defect with all agents.
- 3. *Interaction3*: Cooperate only with agents of the same color, defect with all others.
- 4. *Interaction4*: Cooperate only with agents of a different color, defect with all others.

Obviously the most innovative part of the simulation with respect to the existing models resides in these last two behaviors, which allow a more restrictive form of interaction: a "communitarian" or an "anti-communitarian" attitude. The fitness values are exactly the same as for the simpler original case: cooperation award = 1, defection award = 1 + x, and a cooperator facing a defector gets 0. Again, the value of "x" will be varied in between 0 and 0.25 (between full defection and full cooperation) and again, as the next section will show, the most intriguing and interesting results are obtained for the intermediary range (for which cooperation hardly begins to emerge despite the strong pressure of defection). Seven subbehaviors turn out to be possible but, at the end, every single agent behavior is composed of a pair of one "movement" and one "interaction" sub-behavior (out of twelve possible pairings). Any agent first moves then interact.

The sequential unfolding of the whole simulation can be easily grasped out of the sequence diagram. First, the "world" class asks all agents to act. Then, each agent's action consists in the succession of the two sub-behaviors: first the movement then the interaction. The true movements slightly negatively impact the fitness but the most important fitness contribution is due to the interaction between the agents, in agreement with the classical prisoner dilemma fitness table.

Finally, the agents (while always keeping their color) adapt their type by imitating their most successful neighbors, following the same probability as explained above. When an agent adopts the behavior of its fittest neighbor, it copies both sub-behaviors: the movement and the interaction ones.

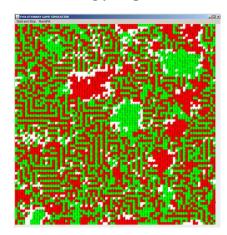
The simulation is run for a given number of time steps, in general until an almost stable configuration (visually assessed) is reached. At the beginning of the simulation, the world is randomly filled with agents of the two colors.

While Novak et al (1994), fascinated by the fractals and kaleidoscopes figures obtained by their simulations, have always tend to favor a deterministic (agents imitate their most successful neighbor) and synchronous form of evolutionary game simulation (all agents play the game then all agents imitate their neighbors), the simulation here adopts instead a stochastic and asynchronous form of updating. In (Bersini and Detours, 1994), it has definitely been advocated why this type of simulation and updating rule should always be preferred over the alternative one. Many irrelevant computational artifact effects are avoided but mainly, it is very hard to conceive natural objects which simultaneously update in time according to a precise central clock.

Experimental results

The experimental results of our simulation will be shown in two forms: first the final board then the evolution in time of the statistical distribution of all seven sub-behaviors among the agents. This second set of dynamical data testifies of the relative evolutionary fitness success of all sub-behaviors and is very reminiscent of the measure of evolutionary activities proposed in (Bedau and Packard, 1992).

1) x = 0 (A strongly cooperative world)



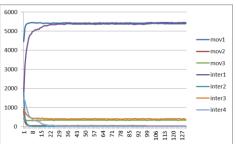
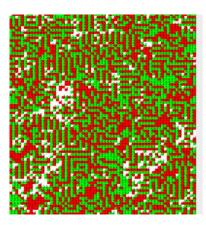


Figure 4: The results in the case of a very cooperative world.

The world remains well mixed and, without surprise, the two winning sub-behaviors are "movement1" (i.e. stay immobile) and "interaction1" (cooperate with all). There is a very timid background activity of the other sub-behaviors that can explain the appearance of tiny local clusters (due to movement3) and "geometrical figures" (due to movement2). While the appearance of clusters is obvious to explain, it is equally easy to understand why the geometrical parallelizing of lines of same color agents is the best way to satisfy the anti-communitarian attitude (maximum of neighbors of different color).

2) x = 0.30 (A strongly defective world)

Again, with no big surprise, the winning sub-behavior is really "inter2" (defect with all). Both "movement1" and a bit of "movement2" (that justifies the appearance of the geometrical configuration) slightly manifest themselves. The reason why the "movement2 (i.e. the "anti-communitarian" displacement) prevails over the opposite one when general defection is the main outcome will be explained in the next paragraph.



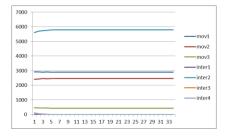
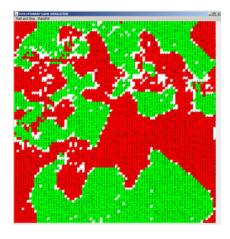


Figure 5: the results in the case of a very defective world.

3) x = 0.10 (An intermediary subtle world)



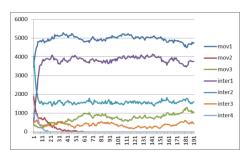


Figure 6: The results in the case of an intermediary world

Such as with the Novak's inspiring original simulation, this intermediary strength of defection leads to the most interesting collective phenomena. Let's discuss the main outcomes. First of all (and this is perhaps the most surprising and interesting effect), a look at the resulting board shows how the world has now become highly segregated. The agents have chosen to cluster according to their color, though no fitness pressure encourages them to behave so (even the contrary). Additionally, the selection of the most adaptive behaviors is much more fluctuating in time than in the previous case due to the continuous competition between defectors and cooperators. The two most successful subbehaviors are "staying immobile" and "cooperate with all", although the third one "defect with all" does not completely vanish. Indeed, such as with the original model (in the absence of any displacement), the intermediary value of the defection pressure leads to a mix of defection and cooperation where for the cooperators to survive they need to self-protect themselves by composing encapsulating clusters.

However, one key difference with the two previous simulations is the presence of two other less successful but nevertheless surviving sub-behaviors: "move3" and "inter3" i.e. change location in the case of a too distinct neighborhood (the communitarian movement) and restrict your cooperative behavior only with agents of the same color (the communitarian cooperation). Moreover, their relative evolutionary importance seems to fluctuate in a very similar way tending to show that they might together characterize the same little ratio of agents. Such as with the Schelling's original model, the fitness success of "move3" is obviously the key reason for the segregation to take place. If new simulations are run, but now in the presence of a fitness penalty for the move much more important, no segregation can take place, while in the case of a less penalized movement this segregation and the success of the "communitarian interaction" are even stronger.

Can we intuitively explain what's going on here and why such segregation occurs although no fitness gain is provided for it (since this movement is penalized).

First, the main result of the Schelling's model is that as a matter of fact only a small ratio of moving agents is enough for the world to become quite segregated. It is clearly the case here since few agents chose to move but with dramatic effects. But, why is it the pair "move3/inter3" that takes the lead and not the alternative pair "move2/inter4" (i.e. the anticommunitarian attitude)? Why the final spatial configuration is filled with clusters and not with geometrical figures? If any agent decides to cooperate only with distinct agents, it is quite simple to understand why its cumulative fitness will be lower than in the communitarian case. Simply, the number of such partners to cooperate with will always be lower in a not segregated world. In other words, a mixed population offers less fitness gain opportunity for "anti-communitarian cooperators" than a very segregated world offers for "communitarian cooperators". When room is left for them to compete, the pairing "move3/inter3" is much more likely to defeat the pairing "move2/inter4".

This same reasoning equally justifies the appearance of the geometrical figures and the relative success of the anticommunitarian attitude in a very defective world (the previous result). This spatial configuration restricts the number of partners each agent can cooperate or defect with in the case of the interaction rules "inter3" and "inter4". For a defector, it is better to defect with less than more (while for a cooperator it is better to cooperate with more than less). Then, one can understand the relative success of the anti-communitarian behavior in a defective world and the relative success of the communitarian attitude as soon as room is left for any form of "reciprocal cooperation".

Figure 7 shows the relative importance of the two movements (communitarian and anti-communitarian) for varying degree of the value 1+x. One can see the initial gradual growing in importance of the communitarian movement as long as cooperation pays and the inversion taking place as soon as the general defective behavior takes the lead (at x = 0.20).

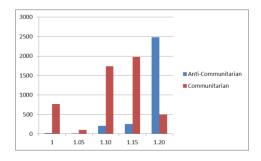


Figure 7: Relative evolutionary success of the anti-communitarian and the communitarian movements ("mov2" and "mov3" above)

The main original result of merging the displacement and grouping strategy of Schelling together with the cooperative/defective evolution of Maynard-Smith/Novak is to facilitate a new route for cooperation favorable to agents which chose to restrict their cooperative attitude to others sharing their same identity.

Conclusions

In a fierce attack against Darwin's natural selection, two philosophers (Fodor and Piattelli-Palmarini, 2006) deny to the classical theory the possibility to distinguish between a trait that has been selectively favored (the heart as a pump) and one that, just by chance, comes together with the previous one. This other trait, although successfully surviving in time, will have never been selected for (e.g. the heart makes noise). Although these attacks have been intensively countered and the book been strongly criticized, it might nonetheless be still valuable to singularize adapted but not per force adaptive traits.

In the model just described, what our simulation results tend to show is that the "segregation behavior" just evolves as a kind of "free-rider" (to re-use the two philosophers terminology) of the "communitarian attitude". This segregating behavior does not appear to be really selected for (since its fitness is even negative) but benefits of the high fitness of the interactive behavior it positively correlates with.

The communitarian attitude is obviously well known to be a definitive very salient trait of human nature. While the Schelling's model never really justified why even this very tolerant regrouping would occur, the simulation discussed in this paper shows that the cooperative gain and the increase in cooperative opportunities (against the prisoner dilemma defective trap) might be the real pressure force that encourage people to assemble according to some distinctive traits (color, religion, social classes, ...).

References

- Bedau, M. and N.H. Packard (1992). Measurements of evolutionary activity, teleology and life. In C.G. Langton, C.E. Taylor, J.D. Farmer, S. Rasmussen (eds.). Artificial Life II. (pp. 431-461). Redwoed, City, California, Addison-Wesley.
- Bersini, H. and V. Detours, 1994. Asynchrony induces stability in cellular automata based models, *Proceedings of the IVth Conference on Artificial Life*, pages 382-387, Cambridge, MA, July 1994, vol 204, no. 1-2, pp. 70-82.
- Bersini, H. (2012). UML for ABM. Journal of Artificial Societies and Social Simulation 15 (1) 9.
- Fodor, J. and Piattelli-Palmarini, M. (2011). What Darwin got wrong. Profile Books.
- Fowler, M. (2003). UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd Edition) Addison-Wesley Professional.
- Maynard-Smith, J. (1982). *Evolution and the theory of games*. Cambridge University Press.
- Nowak, M., Bonhoeffer, S. and R. May (1994). Spatial games and the maintenance of cooperation in *PNAS 91 (11)*: pp. 48-77.
- Nowak, M. A. (2006). Five Rules for the Evolution of Cooperation". Science 314 (5805): 1560–1563.
- Novak, M. (2006). Evolutionary dynamics. Belknap Press.
- Schelling, T.C. (1978). Micromotives and macrobehavior. New-York: Norton.

Social Dynamics and Evolution

Evolution of autonomous hierarchy formation and maintenance

Arend Hintze^{1,2} and Masoud Mirmomeni^{1,3}

¹BEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, MI

²Department of Microbiology

Malandar Control Michigan State University Fact Lansing MI

Molecular Genetics Michigan State University, East Lansing, MI

Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824
hintze@msu.edu, mirmomen@msu.edu

Extended Abstract

Hierarchy among social animals is ubiquitous, and affects the social structures of gregarious species not only by interaction among species within the population, but also through other social forces such as mating, nesting location, amount and the quality of food they receive, or reproductive success. Since T. Schjelderup-Ebbe developed the structural definition of dominance and hierarchy in 1922 (see, e.g., Drews (1993)), different aspects of this social behavior have been addressed. However, exactly how hierarchies can emerge and be maintained among social species is still a conundrum. To investigate this issue, here we analyze a population of autonomous agents ("animates") through the course of evolution. The results of our experiments demonstrate the importance of memory and brain plasticity for the emergence of hierarchy and dominance behavior.

Living in groups has many advantages over living alone. Strength in numbers (Brown, 1996), division of labor (Goldsby et al., 2010), collective hunting (Alexander, 1974), protection via swarming (Olson et al., 2013) are only some of those advantages justifying the evolutionary advantage groups have over solitary living individuals. It has often been remarked that living in groups requires a higher level of organisation than the solitary mode, and provides the driving force for the evolution of more complex cognitive systems in social animals (Ashforth and Mael, 1989). In many cases, hierarchies are used to organize the group, with dominant animals leading and submissive animals falling in line (for a definition of hierarchy see (Drews, 1993)). While social insects like bees or ants solve this organization problem by literally making different organisms (such as queens) to reproduce and workers to perform the labor (Robinson et al., 2008), mammals usually establish their hierarchy using phenotypic traits like strength, aggression, or age (Alexander, 1974; Gauthreaux Jr, 1978). In addition, such hierarchies are usually robust to noise and damage. For example, in case those traits change (or animals die) the submissive animals can become dominant or else animals can reorganize the hierarchy altogether. The ability to robustly reorganize hierarchies requires animals to potentially take on every rank or role in the hierarchy. Here we evolve the capacity to form hierarchies spontaneously (and to reorganize them on the fly) in groups of animats that can communicate on multiple levels (visually, as well as via signalling) with the goal of creating robust collaborative groups. Our animats are controlled by evolvable Markov Gate networks (MGN) (Edlund et al., 2011). Each gate in an MGN is an arbitrary probabilistic (or sometimes deterministic) logic gate, whose function is determined by a set of evolvable probabilities. Here, we introduce in addition gates that can perform simple stack, threshold, counting, and temporal buffering tasks. These MGNs were embodied in a virtual robot (animat) roaming a 2d plane. Animats can turn 90 degrees left and right, move forward, wait, as well as signal a single bit. Sensors allow them to see objects in front of them, the direction it is facing, and to listen to the signals of other animats in their group. Besides these sensors, each animat can use 32 internal bits for computation. We evolve a population of 100 animats

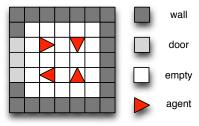


Figure 1: Layout of the test environment (arena). Red triangles: animats. Walls are in dark gray and the door in light gray. The rest of the room is empty (white).

using a classic Genetic Algorithm for 100,000 generations. Animats establish a hierarchy spontaneously by choosing an order they first leave the room. When placed in the same room in subsequent trials, they must leave it in the same exact order. However, these decisions must be made by agents with the same exact brain, that is, the different roles cannot be taken on genetically: they have to be made using information that the agents leave in the arena. The fitness of an animat's genes is determined each by placing four identical copies (clones) of the animat with random orientations in a rectangular room that is enclosed by a wall with a large door on one side of the room (see figure 1). Every time an agent leaves the room by moving through the door, it is removed from the simulation. After the agents have established the hierarchy via the first trial, they are put back randomly to the start locations with random orientations, and accumulate fitness if they leave the room in the same exact order as dictated by the hierarchy in the subsequent 19 trials. Based on the given information, we can write the fitness function as:

$$w = \frac{1}{20 \times 4} \left[\sum_{i=2}^{N} \sum_{j=1}^{4} \delta(v_{ij} - v_{1j}) \right] (1)$$

where $N \leq 20$ is the trail number, v_{ij} is the label of j^{th} robot in the i^{th} round of task, v_{1j} is the label of j^{th} agent the first time agents leave the room, and $\delta(.)$ is Dirac's delta function. Each test is repeated four times for each animat, where the brains are completely reset so that the first order of exits will be random in each test. At the end of each evolutionary experiment, we reconstruct the line of descent (LOD) (Lenski et al., 2003). To test which method of communication was used to establish the order, we re-evaluate the fitness of each agent on the LOD while impeding vision, signalling (or both) by blocking the sensors. The evolutionary experiment was repeated 100 times, and only 11 out of those 100 (from now on called best) evolved a population with a best fitness higher than 0.825 (arbitrary cutoff proportional to getting seven out of eight ranks right), while the average final fitness over all 100 experiments is about 0.625 (Hintze, 2014). This suggests that the task is rather hard to evolve. We find a significant loss in fitness when making

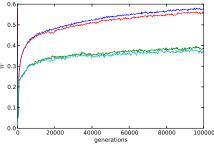


Figure 2: Average fitness (W) on the line of decent over 100.000 generations for all 100 replicates in blue. Fitness for the same bots when they were invisible to each other in green, when they were unable to use their communication channel in red, and when both senses were impaired in least blue.

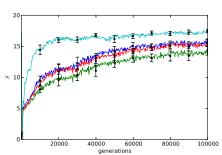


Figure 3: Average success to get individual ranks correct (S) on the line of decent for those eleven best bots. Success that the rank is correct for the first bot in blue, for the second bot in green, for the third bot in red, and for the last bot leaving the room in light blue. In black the standard error.

bots invisible to each other (see figure:2 green), and almost no additional loss in fitness when we block the communication channel (see figure:2 red), indicating that the communication channel is of no important in organizing the hierarchy, and vision alone suffices. It has been argued that the formation of hierarchies requires organisms to evolve submission

first (Alexander (1974)). While in our experiment it is not obvious if bots leaving the room early are dominant over those that leave the room later or the other way around, we find that bots evolve to wait first. When testing the 11 best bots, we find a much better success rate for the last bot leaving the room early on in evolution (see figure:3), than on all other ranks. Only later in evolution the other ranks catch up. We showed that one can evolve virtual clonal bots to form a hierarchy and maintain it over an extensive period. Instead of having distinct properties, these bots were identical and needed to establish an internal representation of a rank through memory and communication. Even though this is a very simplistic environment, we think that this is an important stepping stone towards autonomous self organisation in virtual bots with implications to embedded systems. In the future we plan on studying the robustness of these systems, and how different reward or selection schemes (individual vs. group level selection) effect the formation of hierarchies.

Acknowledgements

We thank Kay Holekamp and Chris Adami for very insightful comments and discussions. This research has been supported in part by the National Science Foundation (NSF) BEACON Center under Cooperative Agreement DBI-0939454. We gratefully acknowledge the support of the Michigan State University High Performance Computing Center and the Institute for Cyber Enabled Research.

References

- Alexander, R. D. (1974). The evolution of social behavior. *Annual review of ecology and systematics*, pages 325–383.
- Ashforth, B. E. and Mael, F. (1989). Social identity theory and the organization. *Academy of management review*, 14(1):20–39.
- Brown, C. R. (1996). Coloniality in the cliff swallow: the effect of group size on social behavior. University of Chicago Press.
- Drews, C. (1993). The concept and definition of dominance in animal behaviour. *Behaviour*, 125(3/4):pp. 283–313.
- Edlund, J. A., Chaumont, N., Hintze, A., Koch, C., Tononi, G., and Adami, C. (2011). Integrated information increases with fitness in the evolution of animats. *PLoS computational biol*ogy, 7(10):e1002236.
- Gauthreaux Jr, S. A. (1978). The ecological significance of behavioral dominance. In *Social Behavior*, pages 17–54. Springer.
- Goldsby, H. J., Knoester, D. B., and Ofria, C. (2010). Evolution of division of labor in genetically homogenous groups. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 135–142. ACM.
- Hintze, A. (2014). evolved brain replicate id 19 all gates. http: //dx.doi.org/10.6084/m9.figshare.987155.
- Lenski, R. E., Ofria, C., Pennock, R. T., and Adami, C. (2003). The evolutionary origin of complex features. *Nature*, 423(6936):139–144.
- Olson, R. S., Hintze, A., Dyer, F. C., Knoester, D. B., and Adami, C. (2013). Predator confusion is sufficient to evolve swarming behaviour. *Journal of The Royal Society Interface*, 10(85):20130305.
- Robinson, G. E., Fernald, R. D., and Clayton, D. F. (2008). Genes and social behavior. *science*, 322(5903):896–900.

REDS: An Energy-Constrained Spatial Social Network Model

Alberto Antonioni¹, Seth Bullock², Marco Tomassini¹

¹Information Systems Department, Faculty of Business and Economics, University of Lausanne, Switzerland ²Institute for Complex Systems Simulation, University of Southampton, United Kingdom

Abstract

The organisation of living systems is neither random nor regular, but tends to exhibit complex structure in the form of clustering and modularity. Here, we present a very simple model that generates random networks with spontaneous community structure reminiscent of living systems, particularly those involving social interaction. We extend the wellknown random geometric graph model, in which spatially embedded networks are constructed subject to a constraint on edge length, in order to capture two key additional features of organic social networks. First, relationships that span longer distances are more costly to maintain. Conversely, relationships between nodes that share neighbours may be less costly to maintain due to social synergy. The resulting networks have several properties in common with those of organic social networks. We demonstrate that the model generates nontrivial community structure and that, unlike for random geometric graphs, densely connected communities do not simply arise as a consequence of an initial locational advantage.

Introduction

The structure of living systems is neither random (where every system element interacts with a random sub-set of other elements) nor regular (where elements interact with neighbours on a lattice). Instead, such systems tend to exhibit complex structure typically featuring clustering and modularity. No doubt much of the detail of this structure arises for reasons that are specific and idiosyncratic to each case. However, self-organisation in simple systems suggests that some characteristic structure may be relatively generic and may arise as a result of fairly simple factors—indeed this type of self-structuring may serve as an important foundation for subsequent evolution and development (e.g., Boerlijst and Hogeweg, 1991; Di Paolo, 2000).

Here we pursue this idea in the context of a social network model (see Toivonen et al., 2009, for an overview of such models). We demonstrate that simple constraints on random network formation due to spatial embedding, limited energy, and the influence of social synergy can generate structures that exhibit key features of social networks: high clustering, right-skewed degree distribution, positive degree assortativity, and strong community structure.

The article is organized as follows. In the next section we give a brief introduction to random geometric graphs, including an energy constrained variant of this network class. Subsequently, we present the REDS model, an extension to the energy-constrained random geometric graph that allows for social synergy to mitigate the costs of maintaining inter-node relationships. The following section presents and discusses the main numerical results, demonstrating that the new model is capable of generating networks that share key properties with real-world social networks. The paper concludes with a summary of the key findings.

Social and Spatial Network Models

Most studies of complex networks, including social networks, consider relational networks where physical distances between nodes are not represented. However, most systems, including social, biological and environmental networks, are embedded in Euclidean space (see Barthélemy, 2011, for a recent review of the field). While relational social network models are important and have been studied in depth (see, e.g., Vázquez, 2003; Catanzaro et al., 2004; Toivonen et al., 2006; Kumpula et al., 2007), their spatial aspects are less well explored (but see Boguñá et al., 2004; Wong et al., 2006; Serrano et al., 2008; zu Erbach-Schoenberg et al., 2014, for some recent attempts).

The canonical spatial network model is the Random Geometric Graph (RGG). We shall use this simple model as the foundation for the work presented here. A RGG is obtained when points located in a plane are connected according to a geometric rule, e.g., connect all pairs of nodes separated by less than a threshold distance, R. There is an extensive mathematical literature on random geometric graphs, particularly in the context of continuum percolation (Dall and Christensen, 2002; Penrose, 2003; Barthélemy, 2011).

In order to generate an N-node RGG with distance threshold, R, distribute N nodes uniformly at random in the unit space, $\Omega \in \mathbb{R}^2$, and add an edge between every pair of nodes separated by a distance r < R, using the standard Euclidean metric on \mathbb{R}^2 . Furthermore, unless otherwise noted, here we shall assume that the unit space, Ω , is the square $[0,1]^2$ with

cyclic boundary conditions (i.e., a torus).

Several RGG variants exist. For example, the Manhattan distance is sometimes used to model mobility networks (Glauche et al., 2003). The general properties of these networks are very close to those employing the more common Euclidean distance, which are the ones we describe here.

The average degree, \overline{k} , of a RGG can be easily estimated as $\overline{k}=\rho V$, where $\rho=N$ is the node density, i.e., the number of nodes per unit space, and V is the neighborhood area. In this case $\overline{k}=N\pi R^2$. The degree distribution of RGGs with a sufficiently large number of nodes can be estimated by the Poisson distribution with parameter $\lambda=\overline{k}$ (Dall and Christensen, 2002).

For large N, the clustering coefficient of a RGG (i.e., the average over all individual node's clustering coefficients, Newman, 2010) tends to $1-\frac{3\sqrt{3}}{4\pi}\sim 0.5865$ for all 2-dimensional RGGs in the Euclidean space (Dall and Christensen, 2002). This important result depends on the particular construction of RGGs. The average clustering coefficient tends to the ratio of the average shared neighborhood area of two connected nodes to the whole neighborhood area. It is clear that changing the radius, R, does not alter this value.

RGGs exhibit positive assortativity, i.e., there is a positive correlation between the degree of pairs of connected nodes (Boccaletti et al., 2006). Antonioni and Tomassini (2012) demonstrate that the assortativity of any *d*-dimensional RGG tends to the value of its average clustering coefficient (a similar result was presented by Barnett et al., 2007, for spatial networks more generally). Many more properties of RGGs are derived by Penrose (2003).

Energy constrained RGGs (Antonioni et al., 2013, hereafter EC-RGGs) are an extension to the standard RGG model where each of a node's connections costs an amount of energy equivalent to its Euclidean length. In addition to the standard constraint that each edge cannot cost more than R, the total cost of an individual node's edges may not exceed some finite threshold value, E. Networks are constructed by assigning legal edges at random until no more edges can be afforded. For large E, EC-RGGs tend to become equivalent to RGGs, saturating such that all edges of length less than R are present in the graph. Where both E and R are large, complete graphs are obtained. However, where both E and R are limiting factors, EC-RGG graphs exhibit a range of clustering and (positive) assortativity values (unlike RGGs). However, neither RGGs nor EC-RGGs exhibit the skewed degree distributions and community structure that are characteristic of social networks.

The REDS Model

The REDS model builds on the RGG and EC-RGG models by including and parameterising the positive influence of shared network neighbours on the cost of maintaining relationships (see Fig. 1). The intuitions here are that (i) there is a limit to the distance over which a relationship can be

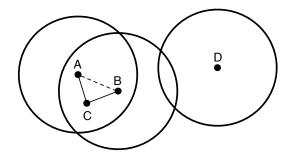


Figure 1: Nodes (dots) may become linked by edges (solid lines) only if (i) they fall within each other's social reach (circles), excluding the possibility of edges AD and BD, and (ii) they can afford to. The cost of edge AB increases with the distance between A and B, but may be reduced by the presence of any shared neighbours of A and B, e.g., C.

maintained, (ii) relationships between nodes vary in terms of cost, (iii) longer distance relationships cost proportionately more than shorter distance relationships, and (iv) relationships with individuals that are themselves connected together may be cheaper to maintain.

This last intuition is exemplified as follows. If I have two friends that know each other, in order to keep my relationship with each of them alive I don't have to physically visit or interact with each of them to the same extent that I would have to in order to maintain two unconnected friends. Direct interaction with one friend effectively involves an element of indirect interaction with friends that we share in common through gossip, chance encounters, group gatherings, etc. In more general terms, this is a local network effect that represents the potential for synergetic or catalytic interactions between the system elements.

The REDS model thus comprises four components:

- 1. **Reach**: an undirected edge, ij, between a pair of nodes, i and j, may only exist if the Euclidean distance between them, D_{ij} , is less than their "social reach", R.
- 2. Energy: each node, *i*, has a finite quantity of "social energy", *E*, that may be spent on maintaining its edges.
- 3. **D**istance: the cost, c_{ij} , of edge ij is proportionate to the Euclidean "social distance", D_{ij} , between i and j.
- 4. Synergy: the cost, c_{ij} , of edge ij varies inversely with the number of network neighbours that i and j share, k_{ij} . This effect is parameterised using $0 \le S \le 1$.

More explicitly, the cost of each edge is calculated as:

$$c_{ij} = \frac{D_{ij}}{1 + Sk_{ij}},$$

where k_{ij} , the number of neighbours shared by i and j, is the cardinality of the intersection between the set of i's neighbours and the set of j's neighbours.

Thus, when S=0 the model reduces to the energy constrained RGG model, with $c_{ij}=D_{ij}$. However, where $0 < S \le 1$, the model incorporates a local network effect that reduces the cost of edges between nodes that have network neighbours in common. Where S=0 each relationship must be maintained independently, whereas for positive S, while maintaining each relationship always involves a non-zero cost, these costs are lower for relationships that involve nodes with shared neighbours. For example, when two friends meet they may discuss or interact with common friends, reinforcing those relationships at a cost less than that of visiting or interacting with all neighbours individually.

The construction process to build an N-node REDS network with social reach R, social energy E, and social synergy S can be summarized as follows:

- 1. A population of N nodes are distributed uniformly at random in the unit square $\Omega \in \mathbb{R}^2$. Each node, i, is allocated the same initial energy, $E_i = E$.
- 2. A node i is picked uniformly at random from the population, and a second node j is chosen uniformly at random from the set of nodes for which the Euclidean distance $D_{ij} < R$.
- 3. An undirected edge between i and j is created only if both nodes have sufficient energy to afford the new set of neighbours that would result. For i, this condition is met if $E_i \geq \sum_x c_{ix}^{+ij}$, where c_{\cdot}^{+ij} denotes the cost of an edge in the updated graph including the new edge ij, and x are the neighbours of i in this updated graph. The same condition must hold for j, mutatis mutandis.
- 4. Steps 2 and 3 are repeated until no more edges can be created according to the linking rule.

The maximum cost that a node, i, may need to pay in order to maintain its k edges occurs when either S=0 (no synergy) or none of i's neighbours are connected to each other, i.e., for each neighbour, j, of i, $k_{ij}=0$. In such a case, the total cost to i is $\sum_j D_{ij}$ which is the sum of all the distances from i to its neighbours. This is appropriate, since the worst case scenario is that a node must pay to maintain each of its relationships independently.

The minimum cost that a node, i, may need to pay in order to maintain its k neighbours occurs for scenarios where S=1 and where i's neighbours form a perfect clique, i.e., for each neighbour, j, of i, $k_{ij}=k-1$. In such a case, the total cost to i is $\sum_j \frac{D_{ij}}{1+k-1} = \frac{1}{k} \sum_j D_{ij}$ which is the average distance from i to one of its k neighbours. This is appropriate, since perfect synergy (maximum S) should not reduce the cost of a set of neighbours to less than the cost of maintaining a relationship with one of them.

Notice that edges are undirected and edge costs are symmetric, with $c_{ij} = c_{ji}$. However, it may be the case that while i can afford a potential new edge, ij, the same edge is not affordable for j as a result of i and j having differing existing edge costs that result in j not having enough remaining energy. Such an edge would not be added to the network, since both of the nodes must be able to afford a new edge connecting them, Notice also that (for S > 0) as edges are added to a network, the cost of both existing network edges and potential new edges may change as a consequence of the creation of new shared neighbours. Thus, even if the number of edges in a graph increases monotonically during construction, the amount of residual energy available to individual nodes (and to the network itself) may sometimes increase (although no node ever has access to more than its initial allocation of energy E_i). Thus, unlike both RGGs and EC-RGGs. the construction of a REDS network may be path dependent.

Results

Figure 2 shows three example networks generated by the REDS model: no synergy (top), high synergy (middle), and no synergy compensated for with increased energy (bottom). All three graphs share the same value of $N=10^3$ and R=0.1, and all exhibit some clustering (the presence of triangle motifs in the network), positive assortativity (high-degree nodes tend to be directly connected to other high-degree nodes at a greater than chance frequency) and community structure (sets of nodes exist, within which pairs of nodes are more likely to be connected to each other than to nodes outside the set).

The no-synergy network (Fig. 2-Top) is sparsely connected and lacks distinctive community structure, whereas a network of equivalent energy but increased social synergy (Fig. 2-Middle) has (i) increased mean degree, because social synergy ensures that some edges become cheaper to maintain, and (ii) stronger community structure, because social synergy ensures that adding edges during network construction tends to reduce the cost of nearby edges involving the same nodes, rather than edges in general. A network with no social synergy but increased energy (Fig. 2-Bottom) is capable of achieving the same mean degree as the high-synergy network, but does not achieve its high clustering, assortativity and heterogeneous community structure.

Figure 3 presents a more comprehensive picture of how the mean degree, mean clustering and assortativity of REDS networks vary with model parameters. Each heat map evidences two sharply defined regimes in the $S \times E$ plane of the model's parameter space¹: the "saturated" regime and the "sparse" regime.

 $^{^{1}}$ We hold N and R constant since they can be thought of as defining a "scale" for the model in terms of node density, N, and the average distance between potential neighbours, 2R/3.

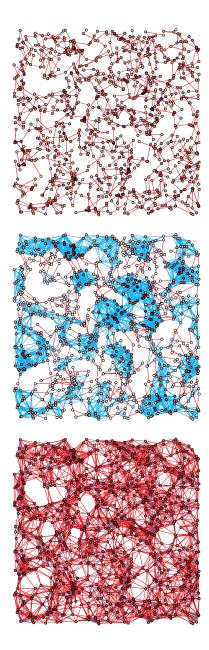


Figure 2: Example REDS networks ($N=10^3,\,R=0.1$). Red nodes have higher degree; red edges have higher cost and blue edges lower cost. $Top~(S=0,\,E=0.15)$: no synergy results in a sparse ($\overline{k}=3$) graph with modest clustering (0.1) and assortativity (0.33). $Middle~(S=1,\,E=0.15)$: maximum synergy results in a dense graph ($\overline{k}=12$) with stronger clustering (0.5) and assortativity (0.65), and evident community structure. $Bottom~(S=0,\,E=0.9)$: no synergy, but sufficient energy to match the middle graph's density ($\overline{k}=13.7$), results in lower clustering (0.34) and assortativity (0.16) and less evident community structure. (Nb. For illustrative clarity, networks were constructed on a bounded unit square, here, rather than a torus.)

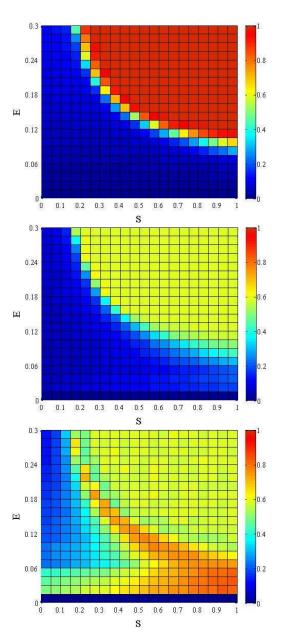


Figure 3: Mean degree (top), clustering coefficient (middle) and assortativity (bottom) for REDS networks ($N=10^4$, R=0.05). Each cell averages 10 independent networks. Mean degree is normalized w.r.t. $k_{max}=N\pi R^2$ (~ 78.54 , the mean degree for a RGG where $N=10^4$ and R=0.05). Clustering is calculated as the mean node clustering coefficient. For large E and/or S, a distinct "saturated" regime exists where degree ~ 78.54 , clustering ~ 0.5865 and assortativity also ~ 0.5865 (i.e., the values predicted for RGGs of equivalent density by Dall and Christensen, 2002; Antonioni and Tomassini, 2012). Outside this regime, degree and clustering are somewhat lower, while assortativity varies considerably with S and E. (For E=0, the empty graph obtains.)

The saturated regime is characterised by high E and high S and is associated with REDS networks that are equivalent to saturated RGGs. This regime corresponds to scenarios in which each node starts with enough energy to accumulate connections to all of the $k=N\pi R^2$ nodes that lie within its social reach. This threshold value for E is high when S=0, because each edge must be paid for independently, but it decreases rapidly as S increases, since synergy reduces the energy that must be spent on edges that close triangles.

Where synergy is minimal (S=0), the total cost incurred by a node, i, when connecting to all of the k nodes within its social reach is $\sum D_{ij} \ \forall j: D_{ij} < R$. Since the mean distance to a neighbour is just 2R/3, the boundary between the two regimes is $E=2N\pi R^3/3$ for S=0 (~ 2.61 for the scenarios plotted in Fig. 3). Where synergy is maximal (S=1), each node still requires some non-zero amount of energy in order to connect to all of the nodes within its social reach. In such a scenario, if a node, i, were able to form a perfect clique with all of the nodes within its social reach, the total cost of i's edges would be equal to $\frac{1}{k} \sum D_{ij} \ \forall j: D_{ij} < R$, i.e., the average distance to a node within its social reach, or 2R/3 (0.03 $\frac{1}{2}$ for the cases plotted in Fig. 3).

However, this value is a lower bound that cannot be reached in practice. First, during the network construction process, before a node can come to be part of minimally expensive clique it must first be part of an incomplete clique that is necessarily more expensive. Therefore, nodes must have access to more energy than is required by the lower bound calculation considered above. It is also the case that the order in which nodes accumulate edges will tend to impact on the extent to which they can achieve a maximal final degree. Second, spatial constraints ensure that a node cannot form a perfect clique with every node within its social reach since some of these nodes will be separated by a distance greater than R and therefore cannot themselves become neighbours. Consequently, for networks where S=1, the regime boundary will tend to occur at E>2R/3.

The saturated regime transitions sharply to a "sparse" regime within which REDS networks are very different from RGG networks. For very low values of E and/or S these networks become fragmented, with nodes unable to afford to maintain more than a few neighbours. However, with moderate values of S and E (below the regime threshold), we find networks that, although sparsely connected by comparison with RGGs, still exhibit significant clustering and a wide range of positive assortativity values, including values that are significantly higher than those of RGGs. This high assortativity indicates the presence of dense pockets of high-degree nodes separated by a hinterland of low-degree nodes connected together (e.g., Fig. 2-Middle).

Moreover, these sparse-regime networks exhibit degree distributions that are very different from the Poisson distributions characteristic of RGGs in the saturated regime (see Fig. 4). Sparse-regime networks tend to exhibit degree distributions that are more sharply peaked than a Poisson and (for some values of S and E) significantly more positively skewed. Interestingly, at and around the boundary between the two regimes, we see degree distributions that are a super-position of the individual distributions associated with each regime, suggesting that while some parts of the network have managed to achieve RGG-like configuration, others have not been able to do so.

More generally, it is instructive to ask: to what extent does a node's final status within the network depend on its initial location in the spatial distribution? Although the distribution of nodes is uniform random, there will necessarily be some variation in the local conditions that each node experiences. Some nodes will have access to more or less potential neighbours within their social reach, R. It might be expected that nodes that are initially disadvantaged by being located in a more sparsely populated patch of space could tend to end up with fewer neighbours in the final graph. Might this effect be responsible for the patches of densely interconnected nodes separated by "hinterland" regions of relatively sparsely connected nodes in some networks?

Figure 5 goes some way towards answering this question by plotting, for the same range of REDS networks displayed in Fig. 4, the final degree achieved by network nodes against the maximum degree that the nodes would have achieved had they been able to connect to every node within their social reach. This figure again reflects the two regimes that we have seen in previous figures. In the saturated regime, the RGG-like networks necessarily exhibit a strong identity relationship between the degree that a node achieves and the maximum degree that it could achieve given the availability of potential neighbours within its reach.

However, for networks within the sparse regime, the relationship between potential degree and actual realised degree is very different. Node degree here is obviously lower in general, but it is also not predicted by the number of potential neighbours within reach. Whether a node is advantaged or disadvantaged by the number of neighbours available at the location in which they are placed has little to do with the degree that they ultimately attain. Indeed, for moderate E and S within this regime (where degree distributions are wider due to the availability of energy and the relative cheapness of some triangle-closing edges) having an average starting location might be most beneficial (e.g., S = 0.5, E=0.09). Again, as with previous figures, we see an interesting hybrid effect at the regime boundary. For instance, where S=0.75 and E=0.09, a large sub-population of nodes exhibit a (slightly depressed) RGG-like distribution, while the remaining nodes are distributed as per a regular sparse regime network. This effect is even more pronounced for S=1 and E=0.09. Again, such scatterplots can be interpreted in terms of hybrid networks within which some spatial regions are close to achieving RGG-like configurations, but other regions are not.

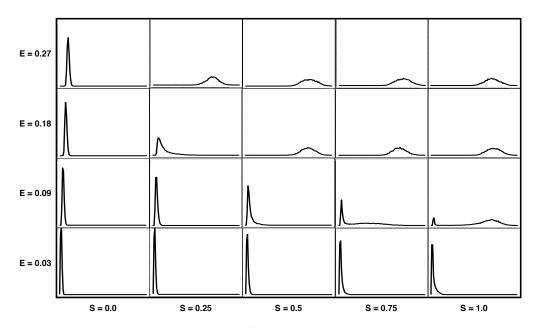


Figure 4: Degree distributions for REDS networks ($N=10^4, R=0.05$). Sub-plot x-axes represent node degree: $0 \le k \le 100$; y-axes represent the number of nodes with that degree: $0 \le p(k) \le 4000$. Two distinctive regimes exist: the "saturated" distribution is Poisson with $\lambda = \overline{k}_{max} \sim 78.54$; the "sparse" distribution is sharply peaked and can be positively skewed. At the boundary between the regimes, hybrid multi-modal distributions can be observed, e.g., for S=1.0, E=0.09.

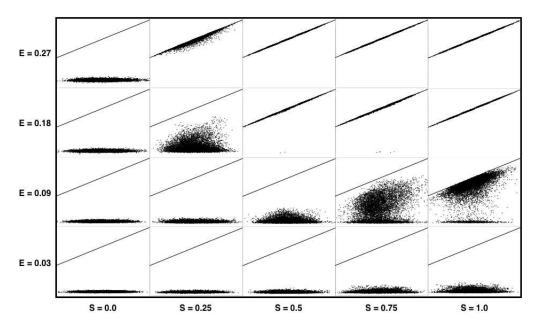


Figure 5: Scatterplots of realised degree (y) against maximum potential degree (x) for REDS networks $(N=10^4,R=0.05)$. Sub-plot x-axes represent maximum potential node degree: $50 \le n \le 110$; y-axes represent actual realised node degree: $0 \le k \le 110$. (A small quantity of jitter noise (<5%) has been added to better indicate density where many datapoints have identical locations.) Again, two distinctive regimes exist: the "saturated" distribution ranges along the line y=x (the upper bound on node degree) with maximum degree predicting realised degree; the "sparse" distribution shows little effect of potential degree on realised degree. Again, hybrid distributions can be observed at the boundary between the regimes, e.g., for S=1.0, E=0.09.

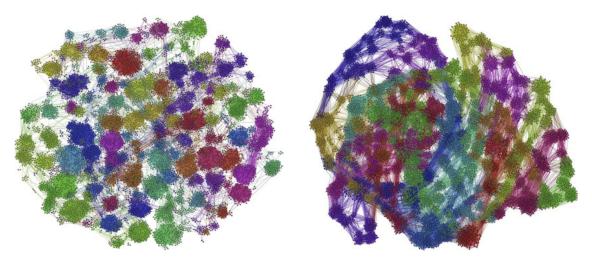


Figure 6: Community structure projections for two REDS networks with differing synergy but similar degree $(N=10^4,R=0.05,\overline{k}\sim 13.5)$. Left (S=1.0,E=0.065): High synergy results in many (49) sharply defined communities (average clustering coefficient = 0.38, modularity = 0.95). Right (S=0.1,E=0.3): Low synergy results in fewer (24) lesswell defined communities (average clustering coefficient = 0.2, modularity = 0.865). Nodes are coloured by community membership. Each layout relocates nodes to reflect their network relationships rather than their original co-ordinates. The OpenOrd algorithm was used for visual representation (Martin et al., 2011). Modularity was calculated using the fast unfolding algorithm due to Blondel et al. (2008). (Nb. For illustrative clarity, networks were constructed on a bounded unit square, here, rather than a torus.)

Figure 6 depicts two REDS networks non-spatially in order to reveal the community structure that they exhibit. Each network is projected onto the 2-d plane in such a way as to reflect its relational, rather than spatial, organisation. Modularity analysis was carried out using the algorithm due to Blondel et al. (2008). By colouring the nodes according to which community they are assigned, we can get a sense of the effect of social synergy on community structure. Whereas the high-synergy network presents a large number of distinct and well-separated communities, the low-synergy network (despite having the same average degree and same spatial distribution of nodes) presents less community structure with fewer resolved communities and more interaction between them. This is consistent with the results presented above, since in the absence of social synergy the uniform distribution of nodes tends to generate an undifferentiated blanket of connectivity, whereas in the presence of social synergy genuine community structure arises and organises in a way that is constrained by distance and energy, but tends to transcend the original spatial layout of the nodes.

Discussion

In the previous section we were able to demonstrate that the number of potential neighbours within a node's reach was not a predictor of its eventual degree for networks within the sparse regime. However, we did not demonstrate what property or properties of a node *did* predict this outcome of the network construction process.

It is likely (although yet to be confirmed) that, in the sparse regime where S>0, whether a node achieves a high or low degree during network construction is determined by the first few edges that are allocated in its locale, rather than the number of potential neighbours within its reach. Nodes that are lucky enough to be assigned edges early in the construction process will enjoy the same kind of rich-get-richer advantage that is enjoyed by nodes that arrive early during a process of preferential attachment (Barabási and Albert, 1999). While preferential attachment explicitly biases network growth in favour of well-connected nodes through its global choice mechanism, the current model achieves something similar by encouraging clique-ish sub-graphs to form around focal nodes with higher than average local clustering.

It is also likely (although yet to be confirmed) that nodes that connect initially to relatively near-by neighbours (rather than other affordable neighbours that are more distant but still within reach) are advantaged during the construction process. Expending energy on connecting with a relatively near-by node means more energy remains to be spent on a new neighbour, and also increases the chance that such a new neighbour can (afford to) close a triangle with you and your first near-by neighbour.

In order to explore this issue, and to better characterise the saturated/sparse regime boundary, it may be useful to consider a "greedy" version of the model in which, rather than picking a random affordable edge, nodes select the cheapest possible new edge.

Conclusion

In this article we have proposed an original model for the construction of social networks that are spatially embedded, constrained by limited energy, and influenced by some degree of social synergy. We started from the random geometric graph model and added three additional ingredients in order to generate networks that possess several of the statistical features exhibited by actual spatial social networks.

The main idea is to attribute a limited but equal amount of social energy to each of a set of spatially embedded nodes. Nodes can spend this resource to link to other nodes as a function of their Euclidean distance, longer links being more expensive than shorter ones, but this cost may be offset by the catalytic or synergetic effect of shared social connections. In this way we obtain networks that resemble realworld networks from the point of view of their statistical features. In particular, the generated networks have high clustering, positive degree correlation, and the presence of community structure. Within a "saturated" regime the model recovers the properties of random geometric graphs, but outside this regime there exists an interesting and varied class of networks that may exhibit degree distributions and community structure reminiscent of organic modular networks.

The model presents several possibilities for further research with the purpose of understanding more about the generic properties of networks inspired by constraints on social processes. For example, one could assume that nodes are not static in space, but move from place to place, perhaps stretching or breaking connections as they do so. This type of process may have the potential to introduce the kind of long-distance links that reduce the characteristic path length of small world networks. Furthermore, it might be reasonable to consider heterogeneity in the distribution of social energy or social reach or social synergy among the nodes. The linking process is bilateral in the present version, i.e., both partners must pay the same amount of energy to create the connection. One-way links could also be considered and the model could be extended to make it dynamical allowing for link removal as well as link formation.

Acknowledgments: We thank the referees for their comments. Bullock was supported by EPSRC grant EP/H021698/1.

References

- Antonioni, A., Egloff, M., and Tomassini, M. (2013). An energy-based model for spatial social networks. In Liò, P., Miglino, O., Nicosia, G., Nolfi, S., and Pavone, M., editors, *Advances in Artificial Life, ECAL 2013*, pages 192–199. MIT Press.
- Antonioni, A. and Tomassini, M. (2012). Degree correlations in random geometric graphs. *Physical Review E*, 86:037101.
- Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286:509–512.
- Barnett, L., Di Paolo, E., and Bullock, S. (2007). Spatially embedded random networks. *Physical Review E*, 76(5):056115.

- Barthélemy (2011). Spatial networks. Physics Reports, 499:1-101.
- Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics*, 2008:P10008.
- Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., and Hwang, D.-U. (2006). Complex networks: Structure and dynamics. *Physics Reports*, 424:175–308.
- Boerlijst, M. C. and Hogeweg, P. (1991). Spiral wave structures in pre-biotic evolution: Hypercycles stable against parasites. *Physica D*, 48:17–28.
- Boguñá, M., Pastor-Satorras, R., Díaz-Guilera, A., and Arenas, A. (2004). Models of social networks based on social distance attachment. *Physical Review E*, 70:056122.
- Catanzaro, M., Caldarelli, G., and Pietronero, L. (2004). Assortative model for social networks. *Physical Review E*, 70(3):037101.
- Dall, J. and Christensen, M. (2002). Random geometric graphs. *Physical Review E*, 66:016121.
- Di Paolo, E. A. (2000). Ecological symmetry breaking can favour the evolution of altruism in an action-response game. *Journal of Theoretical Biology*, 203:135–152.
- Glauche, I., Krause, W., Sollacher, R., and Greiner, M. (2003).
 Continuum percolation of wireless ad hoc communication networks. *Physica A*, 325:577–600.
- Kumpula, J. M., Onnela, J.-P., Saramäki, J., Kaski, K., and Kertész, J. (2007). Emergence of communities in weighted networks. *Physical Review Letters*, 99:228701.
- Martin, S., Brown, W. M., Klavans, R., and Boyack, K. W. (2011). Openord: An open-source toolbox for large graph layout. In *Proc. SPIE 7868*, Visualization and Data Analysis 2011. 786806.
- Newman, M. E. J. (2010). *Networks: An Introduction*. Oxford University Press, Oxford, UK.
- Penrose, M. (2003). *Random Geometric Graphs*. Oxford University Press, Oxford, UK.
- Serrano, M., Krioukov, D., and Boguná, M. (2008). Self-similarity of complex networks and hidden metric spaces. *Physical Review Letters*, 100:078701.
- Toivonen, R., Kovanen, L., Kivelä, M., Onnela, J.-P., Saramäki, J., and Kaski, K. (2009). A comparative study of social network models: Network evolution models and nodal attribute models. Social Networks, 31:240–254.
- Toivonen, R., Onnela, J.-P., Saramäki, J., Hyvönen, J., and Kaski, K. (2006). A model for social networks. *Physica A*, 371:851–860.
- Vázquez, A. (2003). Growing network with local rules: Preferential attachment, clustering hierarchy, and degree correlations. *Physical Review E*, 67:056104.
- Wong, L. H., Pattison, P., and Robins, G. (2006). A spatial model for social networks. *Physica A*, 360(1):99–120.
- zu Erbach-Schoenberg, E., Bullock, S., and Brailsford, S. (2014). A model of spatially constrained social network dynamics. *Social Science Computer Review*. 0894439313511934.

Effects of Evolution on the Emergence of Scale Free Networks

Bijan Ranjbar-Sahraei¹, Daan Bloembergen^{1,3}, Haitham Bou Ammar², Karl Tuyls³ and Gerhard Weiss¹

¹Department of Knowledge Engineering, Maastricht University, The Netherlands

Abstract

The evolution of cooperation in social networks, and the emergence of these networks using simple rules of attachment, have both been studied extensively although mostly in separation. In real-world scenarios, however, these two fields are typically intertwined, where individuals' behavior affect the structural emergence of the network and vice versa. Although much progress has been made in understanding each of the aforementioned fields, many joint characteristics are still unrevealed. In this paper we propose the Simultaneous Emergence and Evolution (SEE) model, aiming at unifying the study of these two fields. The SEE model combines the continuous action prisoner's dilemma (modeling the evolution of cooperation) with preferential attachment (used to model network emergence), enabling the simultaneous study of both structural emergence and behavioral evolution of social networks. A set of empirical experiments show that the SEE model is capable of generating realistic complex networks, while at the same time allowing for the study of the impact of initial conditions on the evolution of cooperation.

Introduction

Understanding the dynamics of networked interactions is of vital importance to a wide range of research areas. For example, these dynamics play a central role in biological systems such as the human brain (Bullmore and Sporns, 2009) or molecular interaction networks within cells (Barabási and Oltvai, 2004); in large technological systems such as the word wide web (Easley and Kleinberg, 2010); in social networks such as Facebook (Backstrom et al., 2011; Ghanem et al., 2012; Ugander et al., 2011); and in economic or financial institutions such as the stock market (Chapman et al., 2012; Jackson, 2008). Recently, researchers have focused on studying the evolution of cooperation in networks of selfinterested individuals, aiming to understand how cooperative behavior can be sustained in the face of individual selfishness (Hofmann et al., 2011; Nowak and May, 1992; Santos and Pacheco, 2005; Ranjbar-Sahraei et al., 2014).

Many studies have targeted the discovery of structural properties of networks that promote cooperation. For instance, Santos and Pacheco (2005) show that cooperation has a higher chance of survival in scale-free networks; Oht-

suki et al. (2006) find a relation between the cost-benefit ratio of cooperation and the average node degree of a network that determines whether cooperation can be sustained; and Van Segbroeck et al. (2010) look at heterogeneity and clustering to find that these structural properties influence behavior on the individual rather than network-wide level. Others have focused on the role of the particular interaction model between neighboring nodes in determining the success of cooperation. For example, Hofmann et al. (2011) simulate various update rules in different network topologies and find that the evolution of cooperation is highly dependent on the combination of update mechanism and network topology. Ranjbar-Sahraei et al. (2014) propose a mathematical model, based on control theory, that allows individuals to choose their actions from a continuous range between pure defection and pure cooperation, and show that this model leads to a higher degree of cooperation than the traditional binary choice models. Control theory is also used by Bloembergen et al. (2014) aiming at ways of influencing the behaviors in social networks.

These studies have assumed the network to be fixed, looking only at the evolution of cooperation over time. In contrast, real-world social networks are not fixed, but continuously change as individuals make and break their ties (Kossinets and Watts, 2006). To this end, Zimmermann and Eguíluz (2005) and Santos et al. (2006) allow individuals to choose with whom to interact, e.g. by giving them the possibility to break ties with 'bad' neighbors and replacing them with a random new connection, and show that such a mechanism may promote cooperation. However, these works still assume a network to be in place, only modifying the connections between nodes over time.

Here, we investigate what happens when nodes are added to the network during interaction. Specifically, we start with an empty network, and add a new node at each time step. Simultaneously, the existing nodes in the network interact following the Continuous Action Iterated Prisoner's Dilemma (CAIPD) model of Ranjbar-Sahraei et al. (2014). New nodes are attached following preferential attachment. Usually, preferential attachment is assumed to follow the

²Computer and Information Science Dept., Grasp Lab, University of Pennsylvania

³Department of Computer Science, University of Liverpool, United Kingdom

Barabási-Albert model (Barabási and Albert, 1999) where links are formed to existing ones proportional to their degree. However, in many social scenarios it intuitively makes sense to look at other individuals' performance rather than their degree when determining with whom to interact - connecting with high performing individuals may give you an edge. We empirically compare both methods of preferential attachment, looking at the structure of the networks formed in detail.

This paper proceeds as follows. First, relevant background is provided on networks and game theory, and an overview of the continuous action iterated prisoner's dilemma (CAIPD) model and preferential attachment is given. These lay the foundation for the proposed Simultaneous Emergence and Evolution (SEE) model that is detailed thereafter. Finally, empirical evaluations highlight the properties of the proposed model.

Background

This section provides background knowledge needed for the remainder of the paper. Firstly, preliminaries on the theory of networks and games are given, constituting the foundation for the model of the evolution of cooperation used in this work. Hereafter, the continuous action iterated prisoners dilemma (CAIPD) is introduced. Finally, preferential attachment, used for the generation of Scale Free (SF) networks, is detailed.

Networks

Networks describe collections of entities (nodes) and the relation between them (edges). Formally, a network can be represented by a graph $\mathbb{G} = (\mathcal{V}, \mathcal{W})$ consisting of a nonempty set of nodes (or vertices) $\mathcal{V} = \{v_1, \dots, v_N\}$ and an $N \times N$ adjacency matrix $\mathcal{W} = [w_{ij}]$ where non-zero entries w_{ij} indicate the (possibly weighted) connection from v_i to v_i . If \mathcal{W} is symmetrical, such that $w_{ij} = w_{ji}$ for all i, j, the graph is said to be undirected, meaning that the connection from node v_i to v_i is equal to the connection from node v_i to v_i . In social networks, for example, one might argue that friendship is usually mutual and hence undirected. This is the approach followed in this work. In general however this need not be the case, in which case the graph is said to be directed, and W asymmetrical. The neighborhood, \mathbb{N} , of a node v_i is defined as the set of nodes it is directly connected to, i.e. $\mathbb{N}(v_i) = \bigcup_j v_j : w_{ij} > 0$. The node's degree $\deg[v_i]$ is given by the cardinality of its neighborhood.

Several types of networks have been proposed that capture the structural properties found in large social, technological or biological networks, two well-known examples being the small-world and scale-free models. The *small-world* model exhibits short average path lengths between nodes and high clustering, two features often found in real-world networks (Watts and Strogatz, 1998). Another model is the *scale-free* network, characterised by a heavy-tailed degree distri-

bution following a power law (Barabási and Albert, 1999). In such networks the majority of nodes will have a small degree while simultaneously there will be relatively many nodes with very large degree, the latter being the hubs or connectors of the network. For a detailed description of networks and their properties, the interested reader is referred to Jackson (2008).

Game Theory

Game theory models strategic interactions in the form of games (Gibbons, 1992). Each player has a set of actions, and a preference over the joint action space that is captured by the received payoffs. The goal for each player is to come up with a strategy (a probability distribution over its actions) that maximizes his expected payoff in the game. A strategy that maximizes the payoff given fixed strategies for all opponents is called a best response to those strategies. The players are thought of as individually rational, in the sense that each player purely tries to maximize his own payoff, and assumes the others are doing likewise. However, this reasoning might not always lead to a beneficial outcome for everyone, and might even be detrimental to all players in the game. Often, there is tension between individual rationality on the one hand, and social welfare on the other.

This archetypal dilemma is aptly captured by the Prisoner's Dilemma (Axelrod and Hamilton, 1981). In this one-shot interaction, players simultaneously choose between either cooperation or defection, after which payoffs are distributed based on their joint action. Cooperation is costly, however cooperators distribute benefits among the other players. Defectors do not pay a cost, but do receive benefits from cooperators as well. In this game, defection (freeriding) is a best response against any opponent strategy, and therefore individually rational players can be expected to defect. However, if all players would cooperate their distributed benefits would outweigh the cost of cooperation, and hence all players would be strictly better off. Herein lies the dilemma.

In this work the players are nodes in the network, repeatedly playing a game with their neighbors. The players have no knowledge of the underlying game, however this repeated interaction allows for adaptation, i.e. to learn a better strategy over time based on the payoff received. The game used in this paper is a generalization of the classical Prisoner's Dilemma, in that the players can have a continuous strategy defining their level of cooperation rather than a binary choice, and payoffs are calculated accordingly. The dilemma, however, remains.

Continuous Action Iterated Prisoner's Dilemma

The continuous action iterated prisoner's dilemma (CAIPD) is a mathematical model of the evolution of cooperation on complex social networks, proposed in Ranjbar-Sahraei et al. (2014). The model describes how the individuals in the net-

work, placed on the nodes, interact with their neighbors according to the aforementioned prisoner's dilemma. Cooperators incur a cost c for each interaction, while their neighbor receives a benefit b with b>c. Defectors free-ride, in the sense that they do not pay costs while still receiving benefits from cooperative neighbors.

Formally, in CAIPD the individuals in a network are represented by N vertices $v_i \in \mathcal{V}$ for $i = \{1, ..., N\}$ on a weighted graph $\mathbb{G} = (\mathcal{V}, \mathcal{W})$. The characteristics of the graph $\mathbb G$ are described by the symmetrically weighted $N \times N$ adjacency matrix $W = [w_{ij}]$. Namely, the connections between the i^{th} and j^{th} individual are denoted by $w_{ij} \in \{0, 1\}$ with all $w_{ii} = 0$. The latter prevent individuals from selfinteraction. In contrast to other existing models, CAIPD allows individuals to choose their level of cooperation from a continuous range between pure cooperation and pure defection, rather than a binary choice between those. This choice is captured by the individuals' state variable $x_i \in [0,1]$. A value of $x_i = 0$ corresponds to pure defection while $x_i = 1$ represents pure cooperation; however x can take on any arbitrary value between those extremes. Extending the prisoner's dilemma to account for this continuous nature of cooperation, a player pays a cost cx_i while the opponent receives a benefit bx_i , again with b > c. This way a defector (i.e., $x_i = 0$) pays no cost and distributes no benefits. Then, the fitness of player i can be calculated as:

$$f_i = -\operatorname{deg}[v_i]cx_i + b\sum_{j=1}^{N} w_{ij}x_j \tag{1}$$

where $\deg[v_i]$ denotes the number of neighbors of the individual v_i . Each player observes how well their neighbors are doing, and will then take over one of their neighbors' strategies with some probability based on their respective fitness difference. In particular, player i adopts the strategy of neighbor j with probability

$$p_{ij} = w_{ij} \cdot \operatorname{sigmoid}(\beta(f_i - f_i)) \tag{2}$$

where $\operatorname{sigmoid}(x) = \frac{1}{1 + \exp(-x)}$, and β is a parameter varying the opponents' influence on individual i.

A network with a state x and topology \mathbb{G} is defined as $\mathbb{G}_x = (\mathbb{G}, x)$ with $x = [x_1, x_2, \dots, x_N]^\mathsf{T}$ representing the state of each node in the network. Such a network \mathbb{G}_x can then be regarded as a dynamical system, where x evolves with respect to time. This evolution depends on a nonlinear mapping

$$\dot{\boldsymbol{x}} = [h_1(\boldsymbol{x}), \dots, h_N(\boldsymbol{x})]^\mathsf{T}. \tag{3}$$

Specifically, the dynamics of the i^{th} player are described by

$$h_i(\mathbf{x}) = \frac{1}{\deg[v_i]} \left[\sum_{j=1}^{N} p_{ij} (x_j(t) - x_i(t)) \right]$$
 (4)

where $\deg[v_i]$ denotes the number of neighbors of player i (i.e., the degree of node v_i); x_i and x_j denote the current cooperation level of players i and j, respectively; and p_{ij} represents the probability with which player i adopts the strategy of player j, as defined in Equation 2. For a detailed description of the CAIPD model, the interested reader is referred to Ranjbar-Sahraei et al. (2014).

The CAIPD model is more general than other existing models, in that is can be used to model the evolution of cooperation on arbitrary complex social networks. Moreover, the continuous nature of the model, allowing for a degree of cooperation rather than a binary choice, is better suited to model many real-world problems (Killingback and Doebeli, 2002). Finally, the deterministic nature of the mathematical model makes it computationally less complex, allowing to study a wider range of scenarios. For these reasons, we adopt this model in the current paper as well.

Preferential Attachment

The preferential attachment model was proposed by Barabási and Albert to explain the power-law degree distribution that is present in many complex real-world networks (Barabási and Albert, 1999). This model is based on the assumption that, in many social settings, the chance of making new connections grows proportionally with the number of connections that you already have (also known as the rich-get-richer phenomenon). The Barabási-Albert model simulates this by growing the network over time, adding one new node at a time, and linking it to a fixed number of existing nodes, these being chosen proportionally to their current degree. Specifically, starting from an initial network of m_0 nodes, at every time step one new node is added to the network. The new node forms $m < |m_0|$ connections to existing nodes, where the probability p_i that the new node connects to existing node v_i is proportional to its degree:

$$p_i = \frac{\deg[v_i]}{\sum_j \deg[v_j]}.$$
 (5)

Preferential attachment generates a long-tailed degree distribution following a *power-law*:

$$P(k) \sim k^{-\alpha}$$

with k denoting the degree of the nodes. The power-law exponent for the Barabási-Albert model is $\alpha=3$; in comparison, many real-works complex networks have been shown to lie in the range $2\leq\alpha\leq4$ (Barabási and Albert, 1999; Newman, 2005).

It is worth noting that although the Barabási-Albert model is a well known model for generating scale-free graphs, individual properties of the nodes other than their degree are typically not taken into account. In real world scenarios, however, both structural as well as behavioral properties affect the preferential attachment process. For instance, in a

co-authorship network, new authors may indeed have a tendency to team up with existing authors that already worked together with many others (i.e., high degree nodes, a structural property), but will also consider paper quality, number of citations, etcetera (i.e., individual behavioral properties).

In this work, we adapt the preferential attachment model of network growth to take behavioral properties into account when forming new links, where the behavior itself follows the dynamics of the CAIPD model. This approach of Simultaneous Emergence and Evolution (SEE) is detailed next.

Simultaneous Emergence and Evolution

Aiming at a unification, the Simultaneous Emergence and Evolution (SEE) model incorporates two evolutionary procedures. The first is concerned with the evolution of behaviors in the network, which follows from the CAIPD model. The second deals with the construction of the network itself. Here, preferential attachment is used. Contrary to previous works, however, the links that each new individual forms with existing nodes depend on the current fitness of those nodes under the CAIPD dynamics, rather than on their degree. Next, an in-depth description of the SEE algorithm is presented.

The SEE Algorithm

Starting from m initially connected individuals, new nodes are added one at a time. The initial state of these m nodes, as well as of each new node, are set randomly to either pure defection or pure cooperation with equal probability. Each new node is connected to m existing ones with a probability proportional to the fitness of the existing nodes, computed according to Equation 1. The connection probability p_i (i.e., the probability that a new node is connected to i) is defined as

$$p_i = \frac{f_i}{\sum_j f_j} \tag{6}$$

where f_i is the fitness of node i and the sum runs over all N pre-existing nodes $j=1,2,\ldots,N$. Therefore, nodes with high fitness tend to quickly accumulate more neighbors, while nodes with low fitness are unlikely to be chosen as the connector for a new node. An upper limit size of $N_{\rm max}$ is defined. This ensures that the network halts its expansion after reaching size $N_{\rm max}$.

In parallel to the structural emergence of the network, the CAIPD model is used to evolve the individual behaviors of the existing nodes. At each iteration, the adjacency matrix \mathcal{W} is updated. Fitnesses are then computed according to Equation 1. These new fitness values are then used to update the state of each node, and therefore of the network as a whole, using the dynamical model of Equations 3 and 4. This is in practice performed with an adequately small step size. The SEE model allows to vary the update rates of both evolutionary processes independently. For example, the behavior of the individual nodes might evolve faster or slower

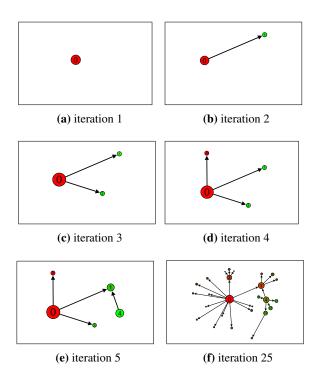


Figure 1: Emergence and evolution of a network according to the SEE model. Node size reflects individual's fitness and its color denotes the state (red for defection to green for cooperation). The direction of arrows shows how individuals influence each other.

than the rate at which new nodes are added. This ratio between the update rate of the behavior and the update rate of the network is defined by $R_{\rm Evo}$, such that when at each time step k a new node is added, the CAIPD model progresses $R_{\rm Evo}$ steps.

Illustration of SEE for a Sample Network

In this section an illustration of the SEE model on a sample network is presented. Initially, there is just one node with pure defection state $x_0=0$, as depicted in Figure 1(a). At the second iteration, Figure 1(b), a cooperating individual is entering the environment (i.e. individual 1), and gets attached to the defector (i.e., individual 0). At this stage, the defector acquires some benefits from the cooperator, while imposing a cost on the cooperator. This results in a higher fitness for the defector than the cooperator (depicted using the node size in the figure).

For further illustration, Figures 1(c)-1(e) show the attachment of three more individuals with defecting or cooperating states (chosen randomly) after joining the network. In parallel to this network emergence, individuals influence each other as described by the CAIPD model, resulting in a simultaneous evolution of their behavior. Figure 1(f) shows the structure and behavioral state of the network after the 25^{th} iteration.

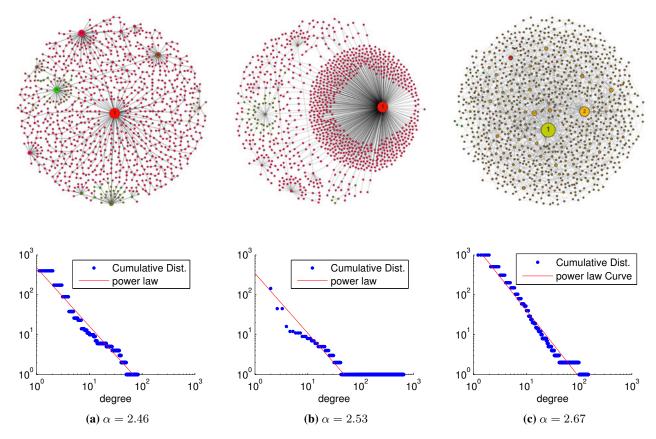


Figure 2: Sample network topologies generated by the SEE model (a) and (b) for m=1 and (c) for m=2, after 1000 iterations. The state and degree of the individuals are denoted by the color (red for pure defection to green for pure cooperation) and size of the nodes. The cumulative degree distribution of each network, shown as blue dots, shows how close this network follows a power law curve, shown as red line, with exponent α .

Experiments and Results

In this section we first illustrate sample networks generated using the proposed SEE model, and show the scale-free characteristics that emerge. Hereafter, the cumulative degree distribution of 8000 different networks generated for 8 different settings of the SEE model will be studied in detail by computing the power law exponent in these networks. Finally, the evolution of cooperation resulting from the proposed SEE model is compared to the standard Barabási-Albert model of preferential attachment. In all experiments, the upper limit for network size is set to $N_{max}=1000$. The number of links added for each new individual, m, is set to either 1 or 2 (indicated where applicable). In the CAIPD model the step size is 0.1; b=4, c=1 and $\beta=1$.

Sample Networks Generated by the SEE Model

Consider an evolution ratio $R_{\rm Evo}$ of 1 in a network that initiates from a single individual which is set initially to either pure defection or pure cooperation. When applying the SEE model, various different network structures can be expected to emerge, as there is stochasticity involved in both initial-

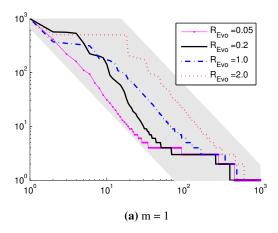
ization of the nodes' states and their attachment. Three samples of such networks are illustrated in the top portion of Figures 2(a)-(c).

In order to study whether the networks generated by the SEE model follow a power law degree distribution, the cumulative degree distribution, i.e., the number of nodes with degree greater than or equal to k, of the sample networks in Figures 2(a)-(c) are shown on a log-log scale¹. The results indeed show a power-law degree distribution with exponent close to $\alpha=2.5$ for the sample networks in Figure 2. Next, we study the average cumulative degree distribution of networks generated by the SEE model in more detail.

Degree Distribution in the SEE Model

In this section, we provide an empirical study on a large number of different networks generated using the proposed SEE model. We analyse different settings for $R_{\rm Evo}$, ranging from 0.05 (slow evolution) to 2 (fast evolution). Figures 3(a)

¹The cumulative distribution of a power law distribution is also power law but with an exponent $\alpha-1$, i.e., $\int cx^{-\alpha}=\frac{c}{1-c}x^{-(\alpha-1)}$, where c is the power law coefficient



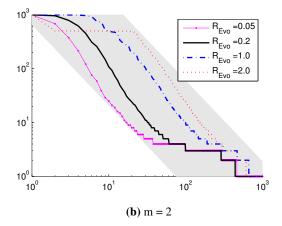


Figure 3: Cumulative Degree distribution of nodes for different evolution rates. The gray region contains the power law distributions corresponding to the scale free networks with exponent $\alpha=2.5$ and different coefficients of the power law.

and (b) show the average cumulative degree distribution of these networks for m=1 and m=2, respectively. For each combination of settings, the SEE model is run 1000 times, with initial nodes randomly set to either cooperation or defection, and the results are averaged.

It can be observed from Figures 3(a) and (b) that the networks that emerged using the SEE model, on average, largely follow a power law degree distribution with exponent close to $\alpha=2.5$. When evolution is slow (i.e., $R_{\rm Evo}\to 0$) the power law is less clearly present, in particular towards the high end of the degree distribution. A possible explanation is that, as the CAIPD evolution slows down, the fitness of the nodes gets updates less frequently as there are fewer interactions. Hence, having more neighbors does not immediately translate to a potential higher fitness.

To get a more detailed insight, the distribution of the exponent of power law distribution that is fit to the constructed networks is illustrated in Figures 4(a)-(d) and 4(e)-(h) for

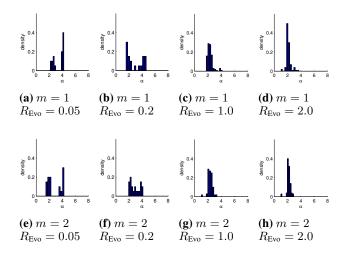


Figure 4: Distribution of the power law exponent for various evolution ratios in the SEE model.

m=1 and m=2, respectively. These figures show that the SEE networks with slow evolution rates exhibit power law degree distribution with exponents $1<\alpha<5$ (i.e., some of the networks fall outside the range of typical real-world complex networks). Increasing the evolution ratio shrinks the range of α values that are observed and centers their distribution around $\alpha=2.5$, yielding realistic scale-free networks. Moreover, it is interesting to note that a bifurcation seems to take place when $R_{\rm Evo}$ decreases (in Figures 4(a), (b) and (e)): the distribution of power law exponents splits into two parts with their mass centered around 2 and 4. This phenomenon warrants a closer inspection in future work.

SEE model vs. Barabási-Albert model

In the previous section the scale-free characteristic of the SEE model was studied and it was shown that the degree distribution of these networks follows a power law degree distribution with $\alpha \approx 2.5$. In this section we study the influence of the SEE model on the evolution of behavior in the network. We compare the proposed SEE model, which uses preferential attachment based on fitness (see Equation 6), with the standard Barábasi-Albert model that uses the degree (see Equation 5). For all experiments, $N_{\rm MAX}=1000$, and the evolution ratio $R_{\rm Evo}$ is set to 1.

Figures 5(a) and (b) show the evolution of cooperation under the SEE model, specifically the figures show the final cooperation level in the network depending on whether the initial nodes where either defectors or cooperators. Similarly, Figures 5(c) and (d) show the same results when the Barábasi-Albert (B-A) model is used for the preferential attachment. It is clear from these figures that the final cooperation level in the network greatly depends on the initial state of the first individuals. When the initial nodes are cooperators, the network tends to cooperate to a large (> 0.5) degree, whereas the situation reverses when the initial nodes

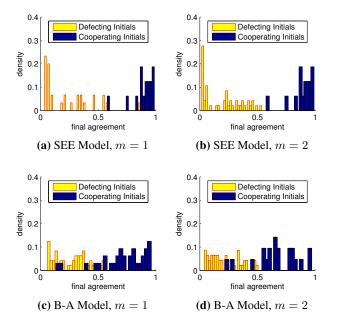


Figure 5: Final degree of cooperation as a function of the initial members' state in the SEE model in (a) and (b) and the Barábasi-Albert (B-A) model in (c) and (d). The colors indicate the state of the initial members: blue for cooperators, and yellow for defectors.

are defectors. This effect is most clear under the SEE model, where a large fraction of the networks eventually reaches either a high (≈ 1) or low (≈ 0) degree of cooperation. When preferential attachment according to Barábasi-Albert is used, this effect is less strong. Here we observe a broad mix of final cooperation levels; moreover the divide between initial cooperators and initial defectors is less clear.

Clearly, the above results demonstrate that final agreements depend on the initial state of the first individuals in the network. This phenomenon is manifested in both the SEE and the Barábasi-Albert model. Having proposed a generalized and formal framework for analysing evolution of cooperation and network emergence, this aspect constitutes a major direction in our future work, where SEE can be used to acquire analytical conclusions describing the effects of such a dependence.

Conclusions

The recent interest to study social networks and their behavior has led to many studies, which can roughly be divided in two streams. The first stream has studied the emergence of these networks, and has in particular tried to find generative models that can explain certain structural properties of real-world complex networks, such as a scale-free degree distribution. The second stream of research has focussed on the evolution of behavior on such networks, when the nodes represent individuals that interact according to some rules. Most notably, interest has been in the evolution of coopera-

tion in social networks, aiming to identify properties of both the network and the interactions that sustain cooperation.

This paper aims to unify these two streams, by studying the simultaneous evolution of behavior on a social network, and the structural emergence of the network itself. The Simultaneous Emergence and Evolution (SEE) model proposed in this paper combines a modified version of preferential attachment, used to generate scale-free networks, with the continuous action iterated prisoner's dilemma (CAIPD) model, describing the evolution of cooperation. Using the proposed model, a number of different networks, emerging from different initial conditions, have been studied. It has been shown that the SEE model yields realistic scale-free networks, despite the fact that the preferential attachment is based on individual's fitness rather than degree.

Moreover, results show that that both structural emergence and behavioral evolution are intertwined, mutually influencing each other, and should therefore be studied in tandem. Aiming at a better understanding of such phenomena, the SEE model provides a fundamental and general framework that allows the analysis of these processes as they coevolve.

An interesting direction for future work is to include the possibility of rewiring as well in the SEE model, whereby existing nodes may break or create links at any time.

References

- Axelrod, R. and Hamilton, W. D. (1981). The evolution of cooperation. *Science*, 211:1390–6.
- Backstrom, L., Boldi, P., and Rosa, M. (2011). Four degrees of separation. *Arxiv preprint arXiv:1111.4570*.
- Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–12.
- Barabási, A.-L. and Oltvai, Z. N. (2004). Network biology: understanding the cell's functional organization. *Nature reviews*. *Genetics*, 5(2):101–13.
- Bloembergen, D., Ranjbar-Sahraei, B., Ammar, H. B., Tuyls, K., and Weiss, G. (2014). Influencing social networks: An optimal control study. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI2014)*.
- Bullmore, E. and Sporns, O. (2009). Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature reviews. Neuroscience*, 10(3):186–98.
- Chapman, M., Tyson, G., Atkinson, K., Luck, M., and McBurney, P. (2012). Social networking and information diffusion in automated markets. In *Joint International Workshop on Trad*ing Agent Design and Analysis (TADA) and Agent-Mediated Electronic Commerce (AMEC), pages 1–14.
- Easley, D. and Kleinberg, J. (2010). Networks, Crowds, and Markets: Reasoning about a Highly Connected World. Cambridge University Press.

- Ghanem, A. G., Vedanarayanan, S., and Minai, A. A. (2012). Agents of influence in social networks. In *Proc. of 11th Int. Conf. on Autonomous Agents and Multiagent Systems (AA-MAS 2012)*.
- Gibbons, R. (1992). A Primer in Game Theory. Pearson Education.
- Hofmann, L.-M., Chakraborty, N., and Sycara, K. (2011). The Evolution of Cooperation in Self-Interested Agent Societies: A Critical Study. Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011), pages 685– 692
- Jackson, M. O. (2008). Social and Economic Networks. Princeton University Press.
- Killingback, T. and Doebeli, M. (2002). The continuous prisoner's dilemma and the evolution of cooperation through reciprocal altruism with variable investment. *The American Naturalist*, 160(4):421–438.
- Kossinets, G. and Watts, D. J. (2006). Empirical analysis of an evolving social network. *Science*, 311(5757):88–90.
- Newman, M. E. (2005). Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46(5):323–351.
- Nowak, M. A. and May, R. M. (1992). Evolutionary games and spatial chaos. *Nature*, 359(6398):826–829.
- Ohtsuki, H., Hauert, C., Lieberman, E., and Nowak, M. A. (2006). A simple rule for the evolution of cooperation on graphs and social networks. *Nature*, 441(7092):502–505.

- Ranjbar-Sahraei, B., Ammar, H. B., Bloembergen, D., Tuyls, K., and Weiss, G. (2014). Theory of cooperation in complex social networks. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-14)*.
- Ranjbar-Sahraei, B., Bou-Ammar, H., Bloembergen, D., Tuyls, K., and Weiss, G. (2014). Evolution of Cooperation in Arbitrary Complex Networks. In *13th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2014)*, pages 677–684.
- Santos, F. and Pacheco, J. (2005). Scale-Free Networks Provide a Unifying Framework for the Emergence of Cooperation. *Physical Review Letters*, 95(9):1–4.
- Santos, F. C., Pacheco, J. M., and Lenaerts, T. (2006). Cooperation prevails when individuals adjust their social ties. *PLoS computational biology*, 2(10):e140.
- Ugander, J., Karrer, B., Backstrom, L., and Marlow, C. (2011). The anatomy of the facebook social graph. *arXiv preprint arXiv:1111.4503*, pages 1–17.
- Van Segbroeck, S., de Jong, S., Nowe, A., Santos, F. C., and Lenaerts, T. (2010). Learning to coordinate in complex networks. *Adaptive Behavior*, 18(5):416–427.
- Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *nature*, 393(6684):440–442.
- Zimmermann, M. G. and Eguíluz, V. M. (2005). Cooperation, social networks, and the emergence of leadership in a prisoner's dilemma with adaptive local interactions. *Physical Review E*, 72(5).

Advancing Social Simulation: Lessons from Demography

Eric Silverman¹, Jakub Bijak¹, Daniel Courgeau² and Robert Franck³

¹University of Southampton, Southampton, UK

²Institut national d'études démographiques, Paris, France

³Université catholique de Louvain, Louvain-la-Neuve, Belgium

e.silverman@soton.ac.uk

Abstract

Previous work has proposed that computational modelling of social systems is composed of two primary streams of research: systems sociology, which is focused on the generation of social theory; and social simulation, which focuses on the study of real-world social systems. Here we argue that the social simulation stream stands to benefit from recent methodological and theoretical advances in demography. Demography has long been an empirically focused discipline focused primarily on mathematical modelling; however, agentbased simulation have proven influential of late as demographers seek to link individual-level behaviours to macro-level patterns. Here we characterise this shift as a move toward system-based modelling, a paradigm in which the scientific object of interest is neither the individual nor the population, but rather the interactions between them. We first describe the four successive paradigms of demography: the period, cohort, event-history and multilevel perspectives. Then we examine how system-based modelling can assist demographers with several major challenges: overcoming complexity in social research; reducing uncertainty; and enhancing theoretical foundations. We propose that this new paradigm can enhance the broader study of populations via social simulation.

Introduction

Understanding the complex and multi-layered processes at play in social systems is far from a simple task, and in recent years research efforts have focused on studying these systems using computer simulation. Taking inspiration from Alife and related areas of complex systems science, the field of *social simulation* has taken shape as a discipline devoted to the study of social systems as complex, non-linear systems composed of interacting agents.

Social simulation has grown steadily over the last decade as social scientists grow increasingly attracted to the idea of representing the complexities of the social realm using simulation (Billari and Prskawetz, 2003). However, questions remain regarding how to best utilise these tools in different contexts. Social simulation offers tremendous flexibility and provides scope for experimentation beyond simple prediction (Epstein, 2008), but what methodological frameworks can guide us in our investigations? Can complex systems

simulation approaches ever be closely linked with traditional empirically-focused social science methods?

Systems Sociology vs. Social Simulation

Previous work has examined the core questions of social simulation in-depth using several case studies as exemplars of different simulation research programmes (Silverman and Bryden, 2007). This work proposed that what we refer to as social simulation can be more accurately represented as two separate streams of research: *systems sociology* and *social simulation*.

Systems sociology encompasses the drive to develop stronger social theories regarding the development and evolution of societies. The use of simulation allows system sociologists to investigate abstract social systems in an effort to understand the core communicative processes that lead to higher-level social complexity; this instantiates the kind of investigations spearheaded by sociological theorists like Niklas Luhmann (1984).

In contrast, social simulation uses computational methods to examine specific elements of social systems, frequently based on real-world examples such as residential segregation (Schelling, 1978), or social care in ageing populations (Silverman et al., 2013). This approach benefits from a closer connection to social phenomena for which usable data is actually available, which can allow for easier validation of simulation results. However, parameterising simulations using the available data is still difficult and is not a straightforward process.

We propose that developing a consistent methodological framework for social simulation is a difficult but necessary task. Simulation approaches allow the modeller to investigate the link between micro-level behaviour and macro-level patterns, which is of great interest to social scientists. In order to further our methodological ambitions, we suggest that social simulation researchers take on board insights from the field of demography, where recent developments have indicated a shift toward a systems-level take on social phenomena that can inform and enhance social simulation efforts.

Examining Demography

Demography as a research field has existed for over 350 years, and during that time has advanced through several successive paradigmatic changes. From the early 'political arithmetics' of Graunt (1662) to current multilevel methods (Courgeau and Franck, 2007), demography has developed increasingly sophisticated means for the examination of population data. The recent popularity of agent-based approaches has prompted a shift in focus toward individual behaviours and interactions and away from strictly population-level insights (Burch, 2003b; Silverman et al., 2011). In this section we will outline the developments in demography over the last three and a half centuries and provide a summary of the primary paradigm shifts that have taken place.

In this discussion we will use a different definition of the word 'paradigm' than those developed by Kuhn (1962). Instead we take inspiration from Granger (1994) and define paradigms as a means to describe the relationship between an observed phenomena and the scientific object. In the case of demography, the scientific object is primarily population-level change – but as we shall see, recent developments indicate a shift in this fundamental perspective.

Courgeau and Franck (2007) identify four such paradigms in demography: cross-sectional (period), longitudinal (cohort), event history and multilevel. Each of these paradigms expands upon and addresses weaknesses of previous paradigms, but we must emphasise that the demography displays *cumulativity* in that new paradigms enhance new avenues of enquiry but do not replace or eliminate previous paradigms. This is similar to the cumulativity we see in other disciplines, such as physics, in which Einstein's relativity addresses weaknesses in Newton's classical mechanics, but nevertheless classical mechanics remains a more than sufficient tool for a great variety of problems in physics.

We note that this perspective on cumulativity can be seen as a departure from Kuhn, who argued that we should "take it for granted that the differences between successive paradigms are both necessary and irreconcilable" (1970). However, following on from Courgeau (2010), we posit that the knowledge generated by successive paradigms displays a non-linear cumulativity. Agazzi has argued that scientific theories focus on a small set of objects, some of which are referential and can differ between theories in the same discipline, while others are contextual and relate to the core concepts which can remain invariant between theories and therefore allow for theory comparison. Agazzi then moves beyond Kuhn and suggests that "scientific progress does not consist in a purely logical relationship between theories, and moreover it is not linear.... and may even be interpreted as an accumulation of truth" (1985).

Within demography, we will see through our historical analysis that the four successive paradigms each surpass certain shortcomings of the previous, and yet these previous paradigms are still in active use. This is because demographers using these different methodologies continue to operate in the same context, even while the perceived *relationships* between demographic events of interest change significantly from one paradigm to another. This suggests that demographic paradigms do not eliminate previous approaches but instead display a non-linear cumulativity of knowledge as new theories sit alongside one another in the broader context (Courgeau, 2010).

Next we will outline the history and structure of demography, and then we will make our case for a fifth paradigm: *system-based modelling*, which is focused on interactions within population systems.

Paradigms in Demography

Demography has its origins in the work of Graunt (1662), who is considered to be the first to apply the scientific method to the study of human populations. Graunt was able to abstract away the concepts of mortality, fertility and morbidity from individuals, and study these as objects of scientific scrutiny in and of themselves. He also connected studies of probability with these examinations of populations, leading to the first studies of events happening to statistical individuals as defined by Courgeau (2012). We characterise these early investigations, and indeed most of the following 200 years of demography, as part of the cross-sectional paradigm, in which population-level events are observed and measured according to historical time. As suggested by Courgeau (2007), we see the cross-sectional paradigm defines the social facts of a given period as existing independently of the individuals affected.

The next paradigm, *longitudinal analysis*, came about through American researchers including Ryder (1951) and was eventually formalised by Henry (1959). Under this paradigm the demographer studies the occurrence of a single event during the life of a cohort (a group of individuals experiencing a particular event during a particular stretch of time). However, this approach requires that this cohort be homogenous and the phenomenon under study must be independent (Courgeau, 2007). The restrictive nature of this approach meant that a new paradigm was soon required in order to allow the study of heterogeneous cohorts and to allow for dependencies between phenomena of interest.

That paradigm soon manifested in the form of the *eventhistory analysis* approach, pioneered by Aalen (1975) following the introduction of general theories of stochastic processes. In this paradigm individuals are seen to follow complex trajectories that depend upon previous events and information they have acquired in the past (Courgeau, 2007). This approach necessitates the use of detailed surveys which can collect data on events and characteristics on an individual level. However, this approach requires significant assumptions when applied to the study of population change; chief among these is the assumption that all individuals are assumed to follow the same random process, the parameters

No.	Paradigm	Period	Key Focus			
1	Period (cross-sectional)	1662-	Population-level phenomena, observed and measured according			
			to historical time			
2	Cohort (longitudinal)	1950s-	Population-level phenomena, observed and measured along the			
			lifetime of individual cohorts			
3	Event History	1980s-	Individual-level phenomena, observed and measured according			
			to individual time			
4	Multilevel	1980s-	Individual, population, and interim-level phenomena, observed			
			and measured from multiple perspectives			
5	System-based	2000s-	Interactions between population systems of individuals, groups			
			and institutions			

Table 1: A summary of the four previous paradigms of demography

of which are estimated from the statistical sample of individuals.

In order to introduce different types of groupings of individuals and allow for the influence of a broader social context, demographers turned to the fourth paradigm: *multilevel analysis*. This context could include socio-economic groupings, social networks, etc. The multilevel approach addresses weaknesses of the event-history approach by allowing for individual behaviour to be constrained by external factors (Courgeau, 2007). While these advantages are significant, multilevel analyses still do not allow for feedback effects (i.e., influences on higher-level behaviour from individual actions), and we shall see how our proposed fifth paradigm addresses these shortcomings.

While we can see a clear progression in demography toward the examination of increasingly complex relationships between social facts at the individual and population levels, each of these paradigms remains in use in certain contexts. Indeed, cross-sectional analyses remain relevant for certain investigations in demography, and this is likely to remain the case for the foreseeable future. In order to examine the complex interactions and feedbacks between the individual level and the population level, however, we need another advancement in methodology. In the next section we outline some of the most pressing challenges facing demography, which will lead to our proposal for the fifth demographic paradigm.

Epistemological Challenges in Demography

The development of demography in recent decades has been closely tied to debates around epistemological challenges. The advent of the multilevel approach, for example, was a product of extensive discussion amongst demographers regarding the examination of multiple levels of analysis in population science. Currently demographers are taking on the issues of uncertainty and complexity.

Demographic changes are inherently uncertain, as with any other aspect of social systems, although demography appears less susceptible to uncertainty in some respects than some related disciplines (e.g., economics). Demographers contend that this is due to the strong empirical slant of the field, and to the power of the statistical relationships which provide the foundations for much of the research in this area (Xie, 2000). Of course some fundamental aspects of population change are considered more uncertain than others; migration, for example, is thought to be much more uncertain than mortality, given that the former can be strongly affected by a number of factors which can change rapidly, such as economic conditions or legislative changes (Xie, 2000). The open and frequent discussion of uncertainty in demography has led to the 'return of the variance' to demography, which has been of great importance to work within all four paradigms described in the previous section (Courgeau, 2012).

The complexity of social phenomena also increases the uncertainty inherent in demographic change, and so additional methodological innovation is required to address these challenges. While significant debate on complexity in demography has been ongoing (Silverman et al., 2011), there is relatively limited evidence available on the performance of demographic models at varying levels of complexity. When looking at predictive applications, opting for simple models that describe uncertainty in detail may be most appropriate (Bijak, 2010). Prediction is far from the only goal in demography, however – as with social simulation (Epstein, 2008) – so other approaches are still required.

These realisations have led to a move toward instantiating demography as a 'model-based science', as we have seen in population biology (Godfrey-Smith, 2006). Some comparisons between modelling in population biology and in artificial life have outlined how these biological applications have important lessons for modellers in other fields (Bullock and Silverman, 2008). However, models of social reality add additional layers of complexity for the modeller; in particular, formalising the relationships between different levels of social phenomena is far from straightforward (Kluver et al., 2003).

Addressing Uncertainty

As these investigations of demographic methodology have progressed, demographers have acknowledged that model uncertainty must be addressed, given the emphasis on formal statistical models in demographic applications. If models themselves are to be acknowledged as sources of additional uncertainty, then the most relevant approach to describing these uncertainties is via Bayesian statistical inference and epistemic probability (see Courgeau, 2012; Raftery, 1995, for discussion). Within this category there are varied approaches to describing model error, from selecting models out of several competing possibilities and related model averaging (Raftery, 1995), to including additional terms for model discrepancy within the modelling process itself (Kennedy and O'Hagan, 2001; O'Hagan, 2006). In addition, Bayesian statistics can allow the modeller to include subjective opinion during the statistical inference process (see, e.g. Bijak, 2010).

Further, Bayesian statistics provides a way to reconcile model-based and empirical approaches by returning to empiricism at a different level of analysis. Computational models, regardless of their structure or complexity, have input parameters and outputs of interest. Various statistical techniques allow for statistical analysis of this mapping, including Bayesian melding (Poole and Raftery, 2000), and Gaussian process emulators (Kennedy and O'Hagan, 2001); for examples of the latter approach in demographic applications, see Bijak et al. (2013) and Silverman et al. (2013). These methods help to alleviate one of the major shortcomings of complex computational models – their relative opacity compared to formal mathematical models – by allowing for an in-depth analysis of the input-output mapping using a formal statistical framework.

Finally, these uncertainty evaluations can be validated by examining various error measures and at empirical frequencies related to predictive intervals with different nominal probabilities. These techniques have been applied in several demographic studies to date, including Bijak (2010), Clark et al. (2012), and Raftery et al. (2012). Quality of calibration can be assessed using scoring rules, as in Gneiting and Raftery (2007).

New Modelling Approaches for Demography

This growing interest in new modelling methodologies for demography has led to a number of proposals in the literature, many of which have been inspired by agent-based computational approaches (see Billari and Prskawetz, 2003; Bijak et al., 2013; Silverman et al., 2013). Demographers have expressed enthusiasm about the possible ramifications for improving the theoretical foundations of demography (Chattoe, 2003; Burch, 2003a), and some recent work has focused explicitly on developing comprehensive social theories based on demographic foundations (Lutz, 2013). Others have pointed out the benefits of using agent-based models

to avoid the over-dependence on increasingly detailed and expensive survey data at the expense of the realism of the explanations offered (Silverman et al., 2011).

The perceived utility of agent-based approaches for explanatory aims has been a significant attraction for demographers (Burch, 2003b; Silverman et al., 2011). Agent-based models are designed to represent the impact of individual behaviours on macro-level patterns and effects. Thus, we contend that agent-based models belong to a broader class of *system-based models*, which are models specifically intended to represent systems composed of interacting elements. In the demographic context, any human population is in fact composed of multiple interacting levels of complexity: individuals, social groups, institutions, etc., any and all of which are suitably complex for in-depth investigation in their own right.

By acknowledging this fundamental aspect of the social realm, demography can continue to shift toward model-based science by making those interactions between different elements of the population an explicit *object of scientific interest*. This then instantiates system-based models in demography as a method for representing interacting, complex behaviours and investigating how these interactions shape demographic change. Re-casting demographic model-building in this way has clear ramifications for future efforts to build stronger theories regarding population change.

This is not to say that system-based approaches are the only option – and indeed, numerous demographic problems will still be addressed perfectly adequately with previous, well-established modelling paradigms. Further, system-based models are somewhat dependent on the presence of sensible theories regarding social systems, and such theories are very difficult to formalise (Kluver et al., 2003; Moss and Edmonds, 2005). A possible way out of this difficulty may be to reconnect system-based approaches to a classical scientific research programme which promotes a functional-mechanistic analysis of populations (Franck, 2002a).

System-Based Modelling

Bringing all of these elements together, we propose that demography is in need of a fifth paradigm, one which maximises the strengths of empirical demographic research as well as the flexibility of simulation methods. While agent-based methods have limitations of their own, as outlined above, this new paradigm integrating these methods into the empirical and inductive research programme of demography may allow us to surpass these difficulties. To do so, we must shift toward a new scientific object in demography: interactions.

Therefore we have proposed a fifth, new paradigm for demography: *system-based modelling*. This approach is based on a functional-mechanistic research programme, which we discuss in more detail in the next section. Agent-based

modelling is of course a possible methodology for implementing a system-based modelling approach, but the overall paradigm need not be tied solely to one methodology. In essence, we propose that demography should seek to examine the interactions between elements of a given population, and the mechanisms driving them. These elements can be described via formal models derived from observation using inductive methods. We posit that this paradigm forms a natural extension of the previous four, and also broadens the scope of demographic research. Following Franck (2002a) and Burch (2003b), we also posit that the system-based approach can enhance the theoretical base of demography by allowing for the development of formal conceptual models.

The strength of simulation-based scenario generation approaches provides one illustration of the potential for this kind of methodological innovation. Demography, as we have seen, has a particular strength in its ability to connect directly to practical, policy-relevant issues (Xie, 2000; Morgan and Lynch, 2001). Methodological enhancements like those outlined above can push this particular strength even further. Bayesian approaches allow for formal statistical decision analysis, and thus support efforts in planning (Bijak, 2010). System-based approaches, when coupled closely with statistical analysis techniques, provides a simulated environment in which policy-makers can investigate a range of possible policy changes. This kind of experimentation is made possible by coherent scenario generation, in which the behaviour of simulated individuals closely follows empirical patterns for statistical individuals observed via demographic data (Courgeau, 2012).

The predictive horizon for demography is thought to be approximately one generation (Keyfitz, 1981), so scenario generation augmented with statistical analysis of model results allows for the exploration of more possible futures (Bijak, 2010). Utilising well-constructed simulation models to investigate these scenarios allows us to examine interactions that drive macro-level patterns of population change in more detail, while also removing some of the limitations of traditional data-heavy statistical methodologies (Silverman et al., 2011). Thus, this application of a system-based approach demonstrates the benefits of an effective combination of the empirical strengths of demography with simulation: demographers can now study complex interactions and behaviours in artificial populations which allow for coherent scenario generation. We have already seen some examples of this type of inductive, empirical system-based approach in the literature – see Klabunde (2014).

A Functional-Mechanistic Research Programme for Demography

Having established the need for, and utility of, a systembased approach in demography, how can we ensure that this new paradigm conforms to a productive classical scientific research programme? We see such an approach extant in

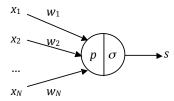


Figure 1: The McCulloch and Pitts (1943) model of a single neuron. Adapted from Franck (2002: 143).

the natural sciences since Francis Bacon. We propose that a move toward system-based modelling as a fifth paradigm presents an opportunity to reconnect with the classical scientific research programme, and in so doing, establish demography as a true model-based science rather than a 'science in the making' (Courgeau and Franck, 2007).

As an illustrative example, consider the famous neural model produced by McCulloch and Pitts (1943), pictured in Figure 1. This model was enormously influential, and led to the development of artificial neural networks some years later. When we look closely at this model, what is actually represented by this simplified neuron?

This model does not represent the physical structure or chemical behaviour of the neuron – we see no representation of the neuron's shape, its genetic material, membrane, etc. What McCulloch and Pitts did was to observe the behaviour of the neuron and represent its functional architecture, without which we would not see its main properties come about. They identified five main functions, which can be seen in Figure 1: receiving input stimuli; weighting inputs with synaptic coefficients; calculating the sum of weighted inputs (p); fixing a threshold of stimulation below which no signal is transmitted; and calculating the exit signal s.

Thus, this neuron model represents the functional structure of the processes that generate the observed behaviour of a neuron. However, the model ignores the causal factors underlying those individual functions – the model is entirely a formal, conceptual construction.

In one sense, McCulloch and Pitts have followed a process similar to reverse engineering: they have induced the design of the neuron from its end products. They of course have no interest in reproducing the neuron physically, and yet they followed much the same process as engineers seeking to reconstruct the design of a device from the final products. In doing so, they have inferred the structure of the essential functions of a neuron based on its behaviour (Franck, 2002b).

This method is essentially derived from the classical programme of scientific research, in which scientists try to in-

fer (induce) the functional structure which rules the process generating the property of interest¹. We refer to this method here as a *functional-mechanistic* one, as it attempts to model the structure of functions that rule a specific mechanism.

When examining social properties, we must similarly model the structure of social functions that drive social processes which generate those properties, and which are necessary to generate those properties. For example, work in demography has established that some combination of fertility, mortality and migration generate variations in population structure. Similarly, the law of supply and demand was inferred from the observation of markets and the social exchanges which generate them. Karl Marx's study of industrial production and its social organisation led him to infer the general structure of functions which drive this process – and in doing so develop the principle of separating labour and capital. These well-known results of social science were all derived from the application of this classical, functional-mechanistic programme of science.

Once we have established the functional structure underlying the process which generates our social property, we may use this as a means to identify and model social factors that have contributed this process. In doing so we may restrict ourselves to the investigation of factors which have contributed directly to the combination of functions we have uncovered. As an example, demographers take mortality, fertility and migration as the functions which define its scientific object – population change – and thus they undertake their investigations on the empirical study of these processes and social factors related to them.

We thus propose that the new system-based paradigm for demography should follow this functional-mechanistic approach. Model-building should follow a process of collecting all relevant empirical information about the social property of interest, which then allows us to uncover the social functions required by this property. Once this functional structure has been modelled, we can then proceed to the modelling of the factors which have contributed to the process which generated this social process, individual agents, institutions, groups, and so on (Franck, 2002a).

In the case of traditional demographic examinations of the key processes of mortality, fertility and migration, we can see clearly that the four previous paradigms are sufficient to answer a great deal of questions related to these processes. They also allow us to identify the presence of important interactions between various elements of human populations that contribute to these processes. Once we turn our investigations directly toward these interactions, rather than the results of these interactions as part of those key processes, then we are shifting our scientific object – and thus we need to follow the system-based approach. Simulation methods

can allow us to represent these interactions explicitly in a social context defined by our best available knowledge regarding the induced functional-mechanistic structure of the social system, and then investigate the potential impact of changes in these interactions.

The Process of System-Based Modelling

The system-based modelling paradigm proposed here brings with it significant advantages: the paradigm expands upon the previous four while incorporating insights from model-based science in other disciplines. Bringing this paradigm forward will require some substantial demands upon the demographic community, however; system-based modelling requires not only new conceptual developments regarding the relationships and interactions between elements of population systems, but also requires the development and use of new tools which are complex and often difficult to build and to analyse.

Thus we propose that the system-based paradigm would take shape as a four-step process, following on from Franck (*General Conclusion* 2002a):

- 1. Observation of the properties of a given population (data)
- 2. Inference of the formal structure implied by these properties
- 3. Using (2) as a guide to investigating social mechanisms that generate properties
- Model-based investigation of formal structures to seek verification with observed data

Note that statistical modelling and uncertainty qualification could serve as a means to infer the formal structures from data in steps (1) and (2).

As we have seen, multilevel modelling in demography has already provided a means to develop a greater insight into certain aspects of complex population systems by demonstrating the existence of interactions between parts of those systems. System-based modelling allows us to move forward once more by allowing us to explore these interactions as the central object of study. Simulations allow us to explore the the social factors that interest us, and generate scenarios which increase our understanding of those factors. These scenarios could represent everything from individual alterations in behaviour at the agent level, to changes in policy at a societal level (as in Silverman et al. (2013)). Using simulations in such a way also allows us to examine the non-linear impact of changes to complex, interacting social systems – a key element in the behaviour of social systems which is not possible to represent using conventional mathematical models.

In practice, system-based population modelling can rely to some extent on the existing agent-based approaches, insofar as they are subjected to the inductive principles of the

¹The type of induction we refer to here is drawn from Francis Bacon, in which induction proceeds similarly to *reverse engineering* as described above. See Franck (2002a) for further explication.

scientific method. However, a key open question remains: what principles should be followed to illuminate the inductive construction of such models? We suggest that this question is fundamental to the future direction of demography, and we have no doubt that vigorous and open debate on this question will be a vital part of efforts to take demography in a new direction.

Lessons for Social Simulation

System-based modelling will present some significant challenges for demographers in order to take advantage of these potential benefits. However, social simulation researchers are by no means exempt from this requirement to adjust their methods in response to the changing landscape in demography. For those of us who wish to use social simulation to investigate real-world populations and their properties, we may expect that the recent developments in demography will have significant impact on our research methodologies.

Looking at the field as a whole, social simulation is a continually growing area of research which continues to establish new links with numerous and varied areas of social science. The attraction of simulation approaches, particularly agent-based models, for social scientists is undeniable; the prospect of simulating individual behaviour and being able to observe the resulting macro-level effects is enticing to not only demographers, but political scientists and economists, among others.

What we have proposed here is admittedly very ambitious – our proposed fifth paradigm has implications for all of social science, in certain respects, and demands a new level of engagement between social simulation practitioners with social scientists. We argue here that the payoff for such engagement will be significant, resulting in stronger theoretical frameworks for social scientists and more empirical relevance for social simulation.

Examining the history of demography has revealed however that such a change would not obviate the need for social simulation as a separate discipline. In demography, the previous four paradigms of research remain very much present, and indeed our proposed fifth paradigm would not be suitable for certain demographic questions that are still best answered using other methods. Likewise, system-based methods may be used as vehicles for broader philosophical investigation of social systems, or as a type of opaque thought experiment (Di Paolo et al., 2000), or for many other purposes beyond prediction or the direct study of population change (Epstein, 2008). Therefore we can expect social simulation to remain and to flourish as a discipline making unique contributions to big questions about sociality in general.

However, successful application of the system-based paradigm in demography will require perhaps more methodological shifts from the social simulation side than is commonly accepted. The application of inductive, empirical methods to the simulation of human populations remains a significant challenge, and requires a different approach more firmly connected to empirical investigation and traditional scientific approaches. We hope that our elucidation of this approach and its potential benefits and pitfalls may help drive these efforts forward, and eventually lead to a productive system-based approach that gives us a greater understanding of population change.

Acknowledgements

Eric Silverman and Jakub Bijak gratefully acknowledge the Engineering and Physical Sciences Research Council (EPSRC) grant EP/H021698/1 Care Life Cycle, funded within the Complexity Science in the Real World theme.

References

- Aalen, O. (1975). Statistical inference for a family of counting processes. PhD thesis, Berkeley: University of California.
- Agazzi, E. (1985). Commensurability, incommensurability, and cumulativity in scientific knowledge,. *Erkenntniss*, 22:51– 77.
- Bijak, J. (2010). Forecasting International Migration in Europe: A Bayesian View. Springer Series on Demographic Methods and Population Analysis, Vol. 24. Dordrecht: Springer.
- Bijak, J., Hilton, J., Silverman, E., and Cao, V. (2013). Reforging the wedding ring: Exploring a semi-artificial model of population for the united kingdom with gaussian process emulators. *Demographic Research*, 29(27):729–766.
- Billari, F. and Prskawetz, A. (2003). Agent-Based Computational Demography: Using simulation to improve our understanding of demographic behaviour. New York: Physica Verlag.
- Bullock, S. and Silverman, E. (2008). Levins and the legitimacy of artificial worlds. *Epistemological Perspectives on Simulation, Lisbon, Portugal.*
- Burch, T. (2003a). Data, models, theory and reality: the structure of demographic knowledge. In Billari, F. and Prskawetz, A., editors, *Agent-Based Computational Demography: Using simulation to improve our understanding of demographic behaviour*, pages 19–40. Physica-Verlag, New York.
- Burch, T. (2003b). Demography in a new key: A theory of population theory. *Demographic Research*, 9(11):263–284.
- Chattoe, F. (2003). The role of agent-based models in demographic explanation. In Billari, F. and Prskawetz, A., editors, *Agent-Based Computational Demography: Using simulation to improve our understanding of demographic behaviour*, pages 41–54. Physica-Verlag, New York.
- Clark, S., Thomas, J., and Bao, L. (2012). Estimates of age-specific reductions in hiv prevalence in uganda: Bayesian melding estimation and probabilistic population forecast with an hivenabled cohort component projection model. *Demographic Research*, 27(26):743–774.
- Courgeau, D. (2007). *Multilevel synthesis: From the group to the individual*. Dordrecht: Springer.

- Courgeau, D. (2010). Paradigmes démographiques et cumulativité. In Walliser, B., editor, *La cumulativité du savoir en sciences sociales*, pages 243–276. Editions de l'EHESS, Paris.
- Courgeau, D. (2012). Probability and Social Science: Methodological relationships between the two approaches. Methodos Series 10, Springer.
- Courgeau, D. and Franck, R. (2007). Demography: A fully-formed science or a science in the making? *Population-E*, 62(1):39–46.
- Di Paolo, E., Noble, J., and Bullock, S. (2000). Simulation models as opaque thought experiments. In Bedau, M., McCaskill, J., Packard, N., and Rasmussen, S., editors, *Proceedings of the Seventh International Conference on Artificial Life*, pages 497–506. MIT Press, Cambridge, MA.
- Epstein, J. (2008). Why model? *Journal of Artificial Societies and Social Simulation*, 11(4):12.
- Franck, R., E. (2002a). The Explanatory Power of Models: Bridging the gap between empirical and theoretical research in the social sciences. Kluwer Academic Publishers.
- Franck, R. (2002b). Computer simulation and the reverse engineering method: Conclusions of part ii. In Franck, R., editor, *The Explanatory Power of Models. Methodos Series vol 1*, pages 141–146. Kluwer Academic Publishers, Dordrecht/Boston/London.
- Gneiting, T. and Raftery, A. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statisti*cal Association, 102(477):359–378.
- Godfrey-Smith, P. (2006). The strategy of model-based science. *Biology and Philosophy*, 21(5):725–740.
- Granger, G.-G. (1994). Formes, opéruptions, objets. Librairie Philosophique Vrin.
- Graunt, J. (1662). *Natural and political observations mentioned in a following index, and made upon the bills of mortality*. The Roycroft, London.
- Henry, L. (1959). D'un probeème fondamental de l'analyse démographique. *Population*, 14(1):9–32.
- Kennedy, M. and O'Hagan, T. (2001). Bayesian calibration of computer models. *Journal of the Royal Statistical Society, Series B*, 63(3):425–464.
- Keyfitz, N. (1981). The limits of population forecasting. *Population and Development Review*, 7(4):579–593.
- Klabunde, A. (2014). Computational economic modelling of migration. *Ruhr Economic Papers*, 471.
- Kluver, J., Stoica, C., and Schmidt, J. (2003). Formal models, social theory and computer simulations: Some methodological reflections. *Journal of Artificial Societies and Social Simulation*, 6(2):http://jasss.soc.surrey.ac.uk/6/2/8.html.
- Kuhn, T. (1962). *The Structure of Scientific Revolutions*. University of Chicago Press.
- Kuhn, T. (1970). *Postscript, The Structure of Scientific Revolutions*. University of Chicago Press.

- Luhmann, N. (1984). Social Systems. Stanford University Press.
- Lutz, W. (2013). Demographic metabolism: A predictive theory of socio-economic change. *Population and Development Review*, 38(Supplement):283–301.
- McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(4):115–133.
- Morgan, S. and Lynch, S. (2001). Success and future of demography: The role of data and methods. *Annals of the New York Academy of Sciences*, 954:35–51.
- Moss, S. and Edmonds, B. (2005). Towards good social science. *Journal of Artificial Societies and Social Simulation*, 8(4):http://jasss.soc.surrey.ac.uk/8/4/13.html.
- O'Hagan, A. (2006). Bayesian analysis of computer code outputs: a tutorial. *Reliability Engineering and System Safety*, 91(10-11):1290–1300.
- Poole, D. and Raftery, A. (2000). Inference for deterministic simulation models: The bayesian melding approach. *Journal of the American Statistical Association*, 95(452):1244–1255.
- Raftery, A. (1995). Bayesian model selection in social research. *Sociological Methodology*, 25:111–163.
- Raftery, A., Li, N., Seveikova, H., Gerland, P., and Heilig, G. (2012). Bayesian probabilistic population projections for all countries. *Proceedings of the National Academy of Sciences*, 109:13915–13921.
- Ryder, N. (1951). The cohort approach: Essays in the measurement of temporal variations in demographic behaviour. PhD thesis, New York: Princeton University.
- Schelling, T. (1978). Micromotives and macrobehavior. W.W. Norton
- Silverman, E., Bijak, J., and Noble, J. (2011). Feeding the beast: Can computational demographic models free us from the tyranny of data? In Lenaerts, T., Giacobini, M., Bersini, H., Bourgine, P., Dorigo, M., and Doursat, R., editors, Advances in Artificial Life, ECAL 2011, pages 747–754. MIT Press, Cambridge, MA.
- Silverman, E. and Bryden, J. (2007). From artificial societies to new social science theory. In e Costa, F. A., Rocha, L., Costa, E., Harvey, I., and Coutinho, A., editors, Advances in Artificial Life, 9th European Conference, ECAL 2007 Proceedings, pages 645–654. Springer, Berlin-Heidelberg.
- Silverman, E., Hilton, J., Noble, J., and Bijak, J. (2013). Simulating the cost of social care in an ageing population. In Rekdalsbakken, W., Bye, R., and Zhang, H., editors, *Proceedings of the 27th European Conference on Modelling and Simulation*, pages 689–695. Digitaldruck Pirrot, Dudweiler.
- Xie, Y. (2000). Demography: Past, present and future. *Journal of the American Statistical Association*, 95(450):670–673.

Pseudo-Static Cooperators: Moving Isn't Always about Going Somewhere

Olaf Witkowski, Nathanael Aubert-Kato

University of Tokyo, Japan olaf@sacral.c.u-tokyo.ac.jp, naubert@is.s.u-tokyo.ac.jp

Abstract

The evolution of cooperation has long been studied in Game Theory and Evolutionary Biology. In this study, we investigate the impact of movement control in a spatial version of the Prisoner's Dilemma in a three dimensional space. A population of agents is evolved via an asynchronous genetic algorithm, to optimize their strategy. Our results show that cooperators rapidly join into static clusters, creating favorable niches for fast replications. Surprisingly, even though remaining inside those clusters, cooperators keep moving faster than defectors. We analyze the system dynamics to explain the stability of this behavior.

Introduction

The problem of the evolution of cooperation has been of interest for a long time. This problem is often tackled by using simple models, such as considering interactions to be a game of Prisoner's Dilemma (PD). Early results in game theory showed that cooperation in the case of well-mixed population was not a given (Axelrod and Hamilton 1981; Smith 1982), yet it is a very common phenomenon in nature.

The PD is a classic two-player "game" in which players are given two options: cooperate (C) or defect (D). The payoffs are such that T>R>P>S, where T stands for Temptation (D versus C), R for Reward (C versus C), P for punishment (D versus D) and S for Sucker's payoff (C versus D). It is also often admitted that 2R>T+S, meaning that cooperating is overall better for the whole system, while defecting is better for the individual. In particular, T>R and P>S means that it is always the best choice for an individual to defect, no matter the strategy of its opponent. In a system where everyone can interact with everyone else, without memory of past games or ways to distinguish opponents, defecting is obviously the best strategy. However, it has been shown that spacial locality helps cooperators survive and even thrive (Nowak and May 1993).

This early work has triggered several lines of investigation, in particular attempts to add movement. While results can be mixed in specific cases (Sicardi et al. 2009), it is widely recognized that movement is helpful (Vainstein et al. 2007). Particular interest has been given to random movement (e.g. Chen et al. 2011; Gelimson et al. 2013). In this case, though, we argue that this movement acts as a way to restrict the neighborhood of specific individuals, thus increasing locality. Diffusion (Vainstein and Arenzon 2014) is another example where the environment is sparse, allowing agents to move to empty areas. Interesting dynamics can also be obtained when the agents can actually choose on their own when and/or where to move (Aktipis 2004, 2011).

In this work, we investigate the impact of limited movement control on agents in a three dimensional space. Agents are all moving at a common constant speed, but choose their direction through the output of a neural network. We also add the possibility to communicate, through the emission of signal. Such communication might be similar to greenbeards, a phenomenon where an otherwise useless phenotype element is used to choose whether to cooperate or not (see for instance Gardner and West 2010). We argue, however, that a slightly different mechanism is at work in our case. Indeed, since the signal is also an output of the neural network, agents can adapt their response to the environment. Signal may be used both to detect where friendly agents are, or as a way to choose a strategy. In this last case, cooperation can arise both from the fact that related agents will have similar signaling (similar to kin selection), or the adaptability of an external agent (mimicry). We show that, when left to their own devices, cooperators will move more than defectors, even though their cluster is static. They also tend to communicate much more than defectors, displaying a complex dynamic to prevent defectors from taking over. We also show that speed matters, as it impacts the radius of the clus-

In the following, we describe the details of the model used in our experiments. Then, we present the proportion of cooperators over time, and compare it to the static case (no movement allowed). We also show other metrics, such as the average displacement over time and the amount of received signal over time. We then analyse those results as well as giving a simple condition on the survival of a cluster before concluding.

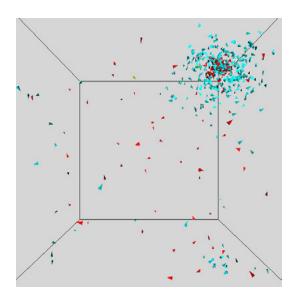


Figure 1: Graphical representation of the world in a simulation. Each agent is represented as an arrow indicating its current direction. The color of an agent indicates its current action, either cooperation (blue) or defection (red). Note the cluster of cooperators being invaded by defectors.

Model

A population of agents move around in a three-dimensional space. Each one is playing the Prisoner's Dilemma game with its direct neighbors. The strategies are evolved via a continuous genetic algorithm, that is agents with high level of fitness are allowed to replicate (with mutations) whenever possible.

Environment

Agents are placed in a three-dimensional world with periodic boundary conditions. While most previous work focuses on two-dimensional simulation, a third dimension gives the system more freedom of movement, making it easier to choose not to play (i.e. move away). The environment is a toroidal cube of size 600 (arbitrary unit), where each face connects directly to the opposite one. The world is considered to be continuous, so that agents can get arbitrarily close to each other (Figure 1), up to the precision of the simulation. Thus, the dimensionality of the simulation comes down to the choice of the agent's interaction radius.

We enforce a maximum size for the population. This makes it easier to compare, for instance, to lattices, where the number of agent also has a physical maximum due to the number of positions. Note that this maximum does not have to be equal to the number of agents at any moment in the simulation. This might also happen in lattices, for instance in Vainstein and Arenzon (2014) where partially empty lat-

tices are used to add a diffusion phenomenon.

Finally, a given simulation is prevented from stopping from lack of agents by adding one new random agent per time step if the current population is below a threshold (see Table 1).

Agents

Agents are given a certain energy, that also acts as their fitness. Each agent comes with a set of 12 different sensors. The neural network (represented on Figure 2) takes the information from those sensors as inputs, in order to decide the agent's actions at every time step. The possible actions amount to the agent's movement, a Prisoner's Dilemma action (cooperate or defect) and two output signals. The architecture is composed of a 12 input, 10 hidden, 5 output, and 10 context neurons connected to the hidden layer (see Figure 2).

The agents' motion is controlled by M_1 and M_2 , outputting two Euler rotation angles: ψ for pitch (i.e. elevation) and θ for yaw (i.e. heading), with floating point values between 0 and π . Even though the agents' speed is fixed, the rotation angles still allow the agent to control its average speed (for example, if ψ is constant and theta equals zero, the agents will continuously loop on a circular trajectory, which results in an almost-zero average speed over 100 steps).

The outputs $S_{\rm out}^{(1)}$ and $S_{\rm out}^{(2)}$ control the signals emitted on two distinct channels, which are propagated through the environment to the agents within a neighboring radius set to 50. The choice for two channels was made to allow for signals of higher complexity, and possibly more interesting dynamics than greenbeard studies (Gardner and West 2010).

The received signals are summed separately for each direction (front, back, right, left, up, down), and weighted by the squared inverse of the emitters distance. This way, agents further away have much less impact on the sensors than closer ones do. Every agent is able to receive signals on the two emission channels, from 6 different directions, totalling 12 different values sensed per time step. For example, the input $S_{\rm in}^{(6,1)}$ corresponds to the signals reaching the agent from the neighbors below.

Fitness

At every time step, agents are playing a N-player version of the prisoner's dilemma with their surrounding, meaning that they make a single decision that affects all agents around them. They get reward and/or punishment based on the number of cooperator around them. Their decision is one of the outputs of their neural network.

The payoff matrix is an extension of Chiong and Kirley (2012), where we added the distance to take into account the

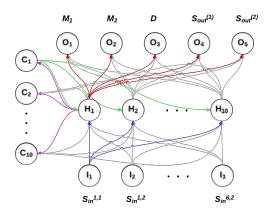


Figure 2: Architecture of the agents controller, composed of 12 input neurons, 10 hidden neurons, 10 context neurons and 5 output neurons.

spatial continuity. It is defined by:

$$\begin{cases} C: b \sum_{\text{coop} \in \text{radius}} \frac{1}{1 + distance(coop, me)} \\ -c \sum_{\text{any} \in \text{radius}} \frac{1}{1 + distance(any, me)} \end{cases}$$

$$D: b \sum_{\text{coop} \in \text{radius}} \frac{1}{1 + distance(coop, me)}$$

$$(1)$$

With b the bonus, c the cooperation cost, b>c>0, and distance the Euclidian distance between two agents. Radius represent the sphere of radius radius around the agent. Note that the agent itself is not considered part of its neighborhood. The distance is not part of the original fitness, which made sense since Chiong and Kirley (2012) are basing their simulation on a lattice, where the distance is always the same. Our version integrates nicely the fact that interactions with distant agents should be much weaker than with closer ones.

Another advantage of this fitness is that defection can also be assimilated to not playing (no cost). Note that there is also no cost and no reward for cooperating when alone.

We can see that this fitness is equivalent to the traditional PD game, since, for two agents A and B at a distance d of each other, (1) yields the payoff matrix:

$$C \quad \frac{C}{1+d} \quad -\frac{D}{1+d}$$

$$D \quad \frac{b}{1+d} \quad 0$$

It is clear that for the conditions b>c>0, this matrix correspond to a PD.

Based on the outcome of the match, agents can choose a new direction, which is similar to leaving the group in the

Initial energy	2
Maximum age	5000
Maximum energy	20
Maximum population size	500
Population threshold	100
Reproduction threshold	10
Reproduction cost	2
Reproduction radius	2
Survival cost per turn	2
Mutation rate (per gene)	0.05

Table 1: Parameters used for the simulation.

walk away strategy (Aktipis 2004), the main difference being that, in our case, it is also possible for groups to split. It is also similar in another aspect: there is a cost to leaving a group, as a lone agent may need time to meet others.

Evolution/Parameters

Evolution is done continuously. Agents with negative or zero energy are removed, while agents with energy above a threshold are forced to reproduce, within the limits of one infant per time step. The reproduction cost is low enough, considering the threshold, to not put the life of the agent at risk. Table 1 indicates the various parameters used for evolution.

Results

Results were obtained on a set of 10 runs, with additional sets used for control. In our setting, all agents have a constant speed, but can choose in which direction they are heading. This allows for pseudo-static behaviors by looping in circles.

While some characteristics, such as agents' movement, were strongly run dependent, the overall dynamics of the system was not. At the beginning of the run, the environment is seeded with random agents. Since all weights in their neural network are set at random, roughly half of the agents initially choose to cooperate while the other half choose to defect. This leads to a fast extinction of cooperators (Figure 3, until approximately 50000 time steps), until a group emerges strong enough to survive. The second phase follows, in which cooperators are quickly increasing in number due to the autocatalytic nature of this strategy (Figure 3).

A third step happens eventually, where defectors invade the cluster, followed either by the survival of the cluster due to cooperators running away or a reboot of the cycle. In case of survival, oscillations in the proportion of cooperators can be observed. However, this phenomenon is averaged away over multiple runs, since period and phase of the oscillations are not correlated from one experiment to the other. Figure 4 shows those oscillations in a typical run. The frequency of those phenomenon is shown in Table 2.

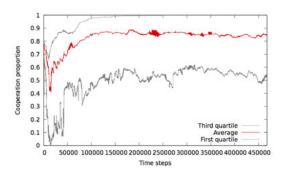


Figure 3: First quartile, average and third quartile of cooperation proportion over 20 runs. Note that agents may choose at each time step which action (cooperation or defection) they will perform, leading to high-frequency noise.

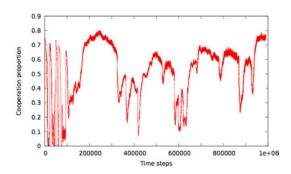


Figure 4: Proportion of cooperating agents in a typical run. Clear oscillations between the "high cooperation" state and the "low oscillation" state are observable.

Minimum	2
First quartile	2.5
Median	4
Third quartile	8
Maximum	9
Average	5

Table 2: Number of oscillations between high and low cooperations over 10^6 time steps in ten runs

As a control, we ran the simulation after removing the possibility for agents to move. In this case, cooperators have much less to fear from defectors and quickly overtake the whole population while defectors quickly exhaust their energy as well as the energy of their cooperative neighbors (Figure 5). Were a defector to appear near a cluster of cooperators, the cluster would react by "reproducing away". However, the chances to be overtaken by the defectors is much higher than in the dynamic case.

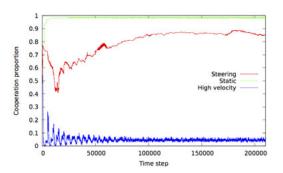


Figure 5: Average proportion of cooperators, comparison between the static and dynamic cases.

Another control was to allow agents to have a neighborhood large enough to interact with all other agents, or a speed such that the system is virtually well-mixed. In both cases, the classical result holds, with an almost homogeneous population of defectors, with the occasional cooperator obtained from random generation.

Finally, we observed the movement tendencies (figure 6) and signal transmission (figure 8) among the two groups of agents. The average displacement is the norm of the total movement over 100 steps (an example for 5 steps is illustrated at figure 7). It is interesting to note that, even though they mostly stay in clusters, cooperators move more than defectors. In the next section, we will attempt to interpret those results.

Analysis

The critical mass necessary for a cooperator to survive can be computed from its surrounding and from the costs of cooperation (Nowak and May 1993). Let us note R the maximum interaction radius, N the total number of agents inside the neighborhood (excluding the cooperator itself), and n the number of other cooperators in the radius. For the cooperator to survive over time, the costs have to exactly balance or be less than the benefits of cooperation. If we assume that agents are homogeneously distributed in the euclidian sphere around our focus, we can rewrite the sum over all surrounding agents weighted by the distance as an integral

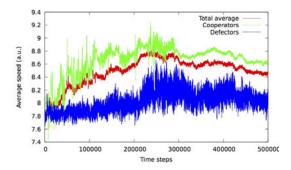


Figure 6: Average displacement of agents over a 100 steps sliding window.

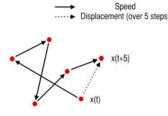


Figure 7: Illustration of the average displacement based on 5 time steps

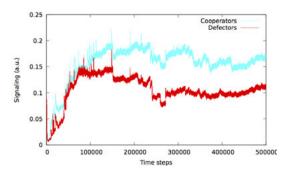


Figure 8: Average signal transmitted by cooperators and defectors.

over the densities ρ_{coop} and ρ_{all} :

$$\rho_{\rm coop} = \frac{3}{4} \cdot \frac{n}{\pi R^3}$$

$$\rho_{\rm all} = \frac{3}{4} \cdot \frac{N}{\pi R^3}$$

This gives us the equivalence:

$$\sum_{\rm coop} \frac{1}{1+dist} \simeq \int_0^R \rho_{\rm coop} \cdot \frac{1}{1+r} dr$$

Which yields:

$$\operatorname{fit_{coop}} = (bn - cN) \, \frac{3\ln(1+R)}{4\pi R^3}$$

Therefore the condition for survival is simply that the proportion of cooperators should be at least $\frac{n}{N} = \frac{c}{h}$.

Note that this condition is strongly dependent on the actual distribution of agents. The closer the cooperators, the stronger they are against external threats. Conversely, a defector at the very center of a group of cooperators can be much more damaging.

In previous work (Chen et al. 2011), it has been observed that random mobility was helping cooperator, if the speed is low enough. However, in this case, this mobility has only the effect of reducing the neighborhood. Additionally, if the speed is too high, the system gets to an almost well-mixed state, with the expected results on cooperation. Note that even the effect of high speed can be counterbalanced by a motion keeping the agents in a neighborhood.

In absence of movement, we have pseudo-movement arising from cooperators dying near defectors. As a result, the cluster of cooperators "reproduces away" from its previous position.

When movement is enabled, cooperators also appear in clusters, inside which they seem to be moving quickly. This mainly results from the major phenomenon helping cooperators, that is their autocatalytic tendencies, which might be a bias from the limit on the population size. If enough cooperators are close to each other, they will keep their energy high at all times, allowing them to reproduce as much as possible. Once the population reaches its maximum capacity, the cooperators typically represent a larger fraction of the population, especially when weighted by the energy they possess. For this reason, the cluster will remain stable until some agents die of old age, before being immediately replaced by other cooperators with a high probability.

Also, this strategy might allow them to avoid spending too much time close to defectors, while remaining constantly in the neighborhood of fellow cooperators.

The clustering is strongly dependent on signaling among the cooperating agents, hinted by the difference in signal emission between cooperators and defectors. Additionally, we performed two batch of five control runs with respectively signal on or off at all time. In the "off" case, no cluster can form, yielding a near-uniform population of defectors. The "on" case still shows qualitatively the emergence of clusters, but are much more diffuse as signaling is now ambiguous.

Conclusion

In this article, we introduced a three-dimensional model of agents playing the Prisoner's Dilemma. While it can be expected that cooperators, if any are present, would quickly evolve to form clusters, it was interesting to see that they still have a higher movement rate overall than defectors. This is even more surprising considering that those clusters do not seem to move fast. Instead, analysis shows that cooperators are moving quickly inside the cluster, which may be a way to adapt to an aggressive environment.

Additionally, comparison with the static case showed that movement made the apparition of cooperators harder, but more stable in the long run. Since it is harder for defectors to overtake a cluster of cooperators, our systems often show a soft bistability, meaning that they will eventually switch from one state to the other. It is even possible to observe a sort of symbiosis, where cooperators are generating more energy than necessary, which is in turn used by peripheral defectors. In this case, replacement rates allow cooperators to stay ahead, keeping this small ecosystem stable.

Finally, this cohesion among cooperators seems to be enhanced by signaling, even though signal might attract defectors. Additional investigation on the transfer of entropy, for instance, could be a promising next step.

Acknowledgements

We would like to thank Julien Hubert for his valuable comments.

References

- Aktipis, C. (2004). Know when to walk away: contingent movement and the evolution of cooperation. *Journal of Theoretical Biology*, 231(2):249–260.
- Aktipis, C. (2011). Is cooperation viable in mobile organisms? simple walk away rule favors the evolution of cooperation in groups. *Evolution and Human Behavior*, 32(4):263–276.
- Axelrod, R. and Hamilton, W. D. (1981). The evolution of cooperation. *Science*, 211(4489):1390–1396.
- Chen, Z., Gao, J., Cai, Y., and Xu, X. (2011). Evolution of cooperation among mobile agents. *Physica A: Statistical Mechanics and its Applications*, 390(9):1615–1622.
- Chiong, R. and Kirley, M. (2012). Random mobility and the evolution of cooperation in spatial n-player iterated prisoners dilemma games. *Physica A: Statistical Mechanics and its Applications*, 391(15):3915–3923.
- Gardner, A. and West, S. A. (2010). Greenbeards. *Evolution*, 64(1):25–38.
- Gelimson, A., Cremer, J., and Frey, E. (2013). Mobility, fitness collection, and the breakdown of cooperation. *Physical Review E*, 87(4):042711.
- Nowak, M. A. and May, R. M. (1993). The spatial dilemmas of evolution. *International Journal of bifurcation and chaos*, 3(01):35–78.
- Sicardi, E. A., Fort, H., Vainstein, M. H., and Arenzon, J. J. (2009). Random mobility and spatial structure often enhance cooperation. *Journal of theoretical biology*, 256(2):240–246.
- Smith, J. M. (1982). Evolution and the Theory of Games. Cambridge university press.

- Vainstein, M. H. and Arenzon, J. J. (2014). Spatial social dilemmas: Dilution, mobility and grouping effects with imitation dynamics. *Physica A: Statistical Mechanics and its Applications*, 394:145–157.
- Vainstein, M. H., TC Silva, A., and Arenzon, J. J. (2007). Does mobility decrease cooperation? *Journal of theoretical biology*, 244(4):722–728.

Invasion of cooperation in scale-free networks: Accumulated vs. average payoffs

Genki Ichinose^{1,2} and Hiroki Sayama²

¹Anan National College of Technology, Anan, Tokushima, JAPAN
²Collective Dynamics of Complex Systems Research Group, Binghamton University, Binghamton, NY, USA ichinose@anan-nct.ac.jp

Abstract

It is well known that cooperation cannot be an evolutionary stable strategy for a non-iterative game in a well-mixed population. In contrast, structured populations favor cooperation since cooperators can benefit each other by forming local clusters. Especially, previous studies have shown that scale-free networks strongly promote cooperation. However, little is known about the invasion mechanism of cooperation in scale-free networks. Here we conducted evolutionary simulations of the evolution of cooperation in scale-free networks where, starting from all defectors, cooperators can spontaneously emerge by mutation. The purpose of this study is to reveal microscopic and macroscopic behaviors of the cooperators' invasion into the network. Since the evolutionary dynamics are influenced by the definition of fitness, we tested two commonly adopted fitness functions: accumulated payoff and average payoff. The simulation results show that cooperation is strongly enhanced in the case of the accumulated payoff when the temptation for defection is low. However, as the temptation becomes higher, cooperation persists more in the case of the average payoff. Moreover, in the case of the average payoff, lower degree nodes play a more important role in spreading cooperative strategies compared to the case of the accumulated payoff.

The emergence of cooperation is one of the challenging problems in both social and biological sciences. Cooperators benefit others by incurring some costs to themselves while defectors do not pay any costs. Therefore, cooperation cannot be an evolutionary stable strategy for a non-iterative game in a well-mixed population. In contrast, structured populations favor cooperation since cooperators can benefit each other by forming local clusters (Nowak and May (1992)). Especially, it is known that scale-free networks strongly promote cooperation (Santos and Pacheco (2005)). However, little is known about the invasion mechanism of cooperation in scale-free networks.

Here we conducted evolutionary simulations of the evolution of cooperation in scale-free networks where, starting from all defectors, cooperators can spontaneously emerge by mutation. The purpose of this study is to reveal microscopic and macroscopic behaviors of the cooperators' invasion into the network. Since the evolutionary dynamics are

influenced by the definition of fitness, we tested two commonly adopted fitness functions: accumulated payoff and average payoff. Barabási-Albert method (Barabási and Albert (1999)) is used for generating initial scale-free networks in simulations. Then, each generated network is substantially randomized by the double-edge swap method while keeping the original degree distributions. Self loops and parallel edges are avoided during the randomization.

A network is made of N nodes occupied by individuals. Each node has its strategy classified as either C (cooperator) or D (defector). Initially, all individuals are defectors. Each node i plays the Prisoner's Dilemma game (PD) with all of its k_i neighbors. The payoffs of the game are calculated as follows. Both individuals obtain R for mutual cooperation and P for mutual defection. If one selects cooperation and the other selects defection, the cooperator obtains S as the sucker of defection, and the defector obtains T as the reward for temptation to defect. The order of the four payoffs is $T > R > P \ge S$ in typical PD. We set P = 0, T = 1 + b, R=1, and S=0, where b>0 is the only control parameter, following previous studies. The payoff of individual iagainst its k_i neighbors is denoted by p_i . Here we consider two types of p_i : accumulated payoff and average payoff. The average payoff is obtained by dividing the accumulated payoff by k_i .

At the beginning of each simulation, one randomly selected individual x plays PD with the neighbors and obtains payoff p_x . Next, one randomly chosen neighbor of x, denoted by y, also plays PD with its neighbors and obtains payoff p_y . If $p_x < p_y$, individual x imitates individual y's strategy with probability $(p_y - p_x)/[(T - S)k_{\max}]$, where $k_{\max} = \max(k_x, k_y)$. Finally, another randomly selected individual z (z might be the same as x or y) flips his strategy (C will become D and D will become C) by mutation with probability m. These operations consist of one time step. We regard N time steps as one generation, in which all individuals are selected once, on average, for the strategy update and mutation. The parameter sets used in the simulations are N=5,000, \bar{k} (average degree) = 4, and m=0.005.

First, we focus on macroscopic behaviors of the cooper-

ators' invasion. Figure 1 shows the fraction of cooperators vs. time (A) and vs. temptation to defect, b (B). As previous studies already revealed, cooperation is promoted in the case of accumulated payoff than the case of average payoff (Fig. 1(A) b=0.2). This is because cooperators on hubs, i.e., highly connected nodes, can gain much greater payoffs than others. However, if the temptation to defect is high (b=0.8), cooperators cannot occupy hub nodes, and therefore the hub effect disappears and cooperation is not promoted, which was not discussed much in previous studies. In contrast, cooperation persists for higher values of b (>0.6) in the case of average payoff, although the fraction of cooperation always stays at low levels (Fig. 1(B)).

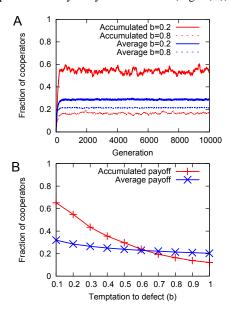


Figure 1: (A) Fraction of cooperators vs. time. Ten independent simulation runs are averaged. (B) Fraction of cooperators vs. temptation to defect (b). Five independent simulation runs (the last 1,000 generations for each) are averaged.

Next, we focus on the microscopic behaviors of cooperator invasion. Figure 2 visualizes simulation results in histograms of propagation events of cooperation, plotted over the degree of the source node of cooperation propagation and its neighbors' state ratio (1 = fully cooperative, 0 = fully defective neighborhood). In the case of the average payoff (Fig. 2(B)), lower degree nodes play a more important role in spreading cooperative strategies compared to the case of the accumulated payoff (Fig. 2(A)), because hubs cannot contribute to the spread of cooperation with the average payoff. Moreover, when cooperators spread, they need a certain fraction of cooperators in the neighbors. However, the fraction of cooperators in the neighbors tends to be low if the degree is high such as hubs. This also makes it difficult for cooperators to spread in the high degree nodes in the case of

the average payoff.

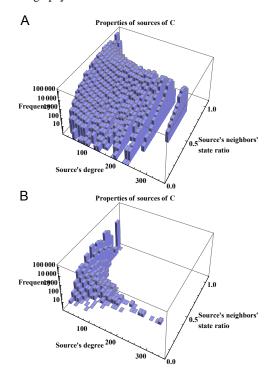


Figure 2: Frequency of propagation events of cooperation, plotted over the source node's degree and its neighbors' state ratio in the accumulated payoff (A) and the average payoff (B) (b=0.2). First 500 generations of 10 simulation runs are accumulated.

In conclusion, we investigated cooperators' invasion in scale-free networks that are initially dominated by defectors. Here we show that cooperation is promoted more in the case of the accumulated payoff only when the temptation to defect is low. This implies that the evolution of cooperation on a network depends significantly on how game players are rewarded through their game play. Moreover, from the indepth analysis of microscopic behaviors, we show that the relative importance of low degree nodes for the evolution of cooperation is much higher in the case of the average payoff, compared to the previously studied case of the accumulated payoff where hubs are known to play a major role in the propagation of cooperation.

References

Barabási, A. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286:509–512.

Nowak, M. A. and May, R. M. (1992). Evolutionary games and spatial chaos. *Nature*, 359:826–829.

Santos, F. C. and Pacheco, J. M. (2005). Scale-free networks provide a unifying framework for the emergence of cooperation. *Physical Review Letters*, 95:098104.

An Agent-Based Model of Indirect Minority Influence on Social Change

Jiin Jung¹ and Aaron Bramson²

¹Claremont Graduate University ²Riken Brain Science Institute jiin.jung@cgu.edu

Abstract

The present study investigates how local majority and minority influences, in combination with an internal consistency process, affect cultural group formation and social change at a global level. We constructed an attitude updating algorithm based primarily on context/categorization-leniency contract theory. This theory postulates when, why, and how people are influenced by an ingroup majority to change an attitude (direct majority influence), by ingroup minorities to immediately change a related attitude (indirect minority influence), and eventually to change attitudes via an internal consistency process. These rules of social influence have been empirically validated in the field of social psychology. However, it is unknown how social influence processes following these rules at a local level lead to larger attitude group formation and social change (a process whereby a nascent opinion becomes the prevailing opinion) at a global level. The present study aims to fill this gap. With minimal assumptions, we implemented our social influence algorithm in an agent-based model to explore how majority and minority influences - along with internal consistency processes - each contribute to cultural group formation and social change. Our results reveal that persistently diverse attitude groups can emerge when minority and majority influences operate together; i.e., internal consistency is not a necessary condition, however it does facilitate attitudinal diversity and maintains it longer. Furthermore, even in the face of the direct majority influence, social change can occur via the indirect minority influence process when combined with internal consistency. We start here with a minimal model, but discuss directions for future expansions.

Theoretical and Empirical Foundations

Social change is the process whereby a society adopts a new belief ¹ which eventually becomes accepted as a norm. Diverse subpopulations are often seen as challengers to social stability and frequent catalysts of social change (Moscovici, 1976). Social psychologists have well-documented why, how, and when people can be influenced by such subpopulations. Furthermore, there are various theories on how the minority influence process can lead to social change (Mucchi-Faina, Pacilli & Pagliaro 2010); but little concrete research has been done to validate these postulates. A primary difficulty lies in the micro/macro differences in the phenomena: social change

happens at the level of society, whereas the minority influence process happens at the interpersonal and intra-individual levels. The present study aims to clearly establish the link between the minority influence process at a local level and social change at a global level by using an agent-based model. The agent-based model fosters examining cross-level links between agents' local behavioral rules and their generated global patterns. The model is primarily based on context/categorization—leniency contract (CCLC) theory (Crano 2010), a theory of majority and minority influences in the field of social psychology.

Minority Influence

Identifying the underlying processes fostering the prevalence of diverse subpopulations and nascent ideas has been among the dominant puzzles of the social sciences. A substantial collection of studies about conformity and majority influences (Sherif 1936; Newcomb 1943; Asch 1951, 1952, 1955; Crutchfield 1955) could not adequately address the issue until Moscovici (1976, 1980, 1985; Moscovici & Faucheux 1972) took the first step to studying the power of minority influence. Since then, research on minority influence has explicated conditions under which a nascent attitude can permeate a population to displace a prevailing attitude on the same issue. conditions include the nascent-idea-holding subpopulation's social identity (ingroup vs. outgroup), the attitudes' composition (consistent vs. inconsistent), the issue objectivity (objective vs. subjective), immediacy (immediate vs. delayed) and the focus of issue being influenced (directly to focal attitude vs. indirect influence) (Crano & Seyranian 2009). In this section, we review Moscovici's conversion theory, and illustrate how Crano's CCLC theory is developed to tighten certain aspects of conversion theory. In the following section we integrate the two theories to construct an algorithm of minority influence.

Moscovici's conversion theory. Moscovici's conversion theory contrasts the differences underpinning the motivational and cognitive processes between majority and minority influence. When individuals are confronted with an attitude counter to their own (whether the prevailing attitude or a nascent one) it creates inner conflict and tension, which results in a motivation to reduce this tension. The psychological discomfort is resolved via different processes depending on whether the counter-attitudinal source is the majority or minority attitude in their ingroup: minority

 $^{^{1}}$ In this paper we use the terms 'belief', 'attitude', 'idea', 'opinion', etc. interchangeably to refer to individuals' epistemic relationship to an issue or topic.

influence occurs through a validation process whereas majority influence occurs through a comparison process. Because the social majority may control resources or other sources of power, adopting the prevailing focal attitude is rewarding. Therefore individuals may change attitudes to comply with the group's consensus by simply comparing their own attitude to the prevailing one for some focal issue. This compliance reflects motives of group belongingness, but not an evaluation of the social majority's arguments for their positions. Consequently, the comparison results in an immediate direct influence, but the attitude change is typically superficial and may easily be changed again.

On the other hand, when someone in the society breaks the attitudinal unanimity, the person's ingroup members take notice of this salient event and the nascent idea captures the other ingroup members' attention. Even though nascent belief holders' positions are not conceived of as correct, their arguments are considered closely by the other ingroup members. At this stage of the validation process, the other ingroup members scrutinize the ideas in-depth. As a consequence, belief validation results in indirect influence (i.e., attitude changes happen in other, non-focal dimensions related to the nascent idea) and/or delayed focal attitude changes. Because ingroup members carefully evaluate the uncommon ideas, attitude changes induced by minorities via this path are stronger and last longer than those by unreflective conformity to the prevailing attitude.

According to conversion theory the most important condition for attitudes held by a minority of group members to influence other ingroup members is attitudinal *consistency*. Moscovici verified consistency's importance in his blue-green study (Moscovici 1969; 1980). Specifically, minority opinions are only influential when their attitudes are perceived as consistent and coherent (Clark 1990; Moscovici, Lage & Naffrechoux 1969; Wood, Lundgren, Ouellette, Busceme & Blackstone 1994).

Crano's context/categorization-leniency contract theory. Later research elaborated on the conditions under which attitudes held by a minority of ingroup members can have an influence on the rest of the group. This influence starts with changes to individuals' attitudes within the local ingroup, and eventually can percolate through the society and lead to global-level social changes in which the nascent idea replaces the prevailing one (Crano 2010; Martin & Hewstone 2001; 2010).

Crano's CCLC theory postulates that when ingroup members offer novel ideas that are not threatening to the ingroup they may be influential. His theory states that due to the leniency contract ² between majority and minority opinions within a group, minority influences lead to both *immediate indirect attitude changes* (i.e., changes to attitudes besides the focal countervailing attitude) and also *delayed focal attitude changes* via an internal consistency process. By contrast, majority influences lead to immediate changes to

the focal attitude, but do not affect supporting non-focal attitudes.

Thus, even though a direct change in the focal attitude is unlikely, there exists pressure to change indirect/related attitudes within the same cognitive constellation. Minority influence then becomes a function of message quality. A weak message may not lead to delayed focal change but rather temporary indirect change may occur in the direction of the gist of message. A strong message leads to an immediate indirect change, but an immediate focal change is unlikely. This minority influence toward indirect attitude changes has been empirically supported (Alvaro & Crano 1997; Crano 2000; Crano & Alvaro 1998; Crano & Chen 1998, Martin & Martin 2006).

Finally, indirect attitude changes may eventually lead to a focal attitude change via internal consistency processes. Because attitudes do not exist in isolation but rather are structurally interrelated in belief constellations, attitudes that occupy the same cognitive constellation may all be affected when one element of the set is changed. As indirect changes accumulate, delayed focal change can occur due to the motivation to maintain internal consistency within the cognitive constellation. (Crano & Chen 1998, p.1440; Fink & Kaplowitz 1993; Judd, Drake, Downing & Krosnick 1991; McGuire 1990; McGuire & McGuire 1991).

The Agent-Based Model

First we present a modeling framework that matches the social psychology theories. We developed an updating algorithm for agents that captures the majority and minority influence processes as well as internal consistency (Figure 1). In the model, we assume N agents indexed by j. Each agent can be represented by a vector of M attitudes that take one of A values; $a^j = [a^j_{\ i}] = a^j_{\ i}, a^j_{\ 2}, \ldots, a^j_{\ M}$, where each $a^j_{\ i} \in \{0,1, \ldots A\}$. We adopt the convention that agent j is the one being influenced and an agent k is the source agent who influences agent j. The source agent has attitudinal consistency if the source's attitudes take the same values (i.e., $a^k_i = a^k_{i+1} = \ldots = a^k_{\ M}$).

First each agent identifies its *local ingroup*; the collection of agents with the potential to influence it. Then each agent randomly selects one member of its local ingroup to be an influence *source*. Each agent then compares a randomly selected *focal attitude* with that of its source agent. If the corresponding attitude of the source matches that of the agent (i.e., $a^i_i = a^k_i$), social influence is not initiated because the source affirms the agent's view (Steele, 1988; Sherman & Cohen, 2006). However, if the corresponding attitudes between the agent and the source differ (i.e., $a^i_i \neq a^k_i$), this discrepancy draws the agent's attention and evokes inner conflict (Festinger 1954; 1957). This motivates the agent to react. The reaction's form is contingent upon the majority/minority status of the source's attitude in the agent's local ingroup (Crano & Seyranian 2009).

If the focal attitude of the source is the majority attitude in the agent's local ingroup, then the agent conforms to the source (i.e., changes that attitude to match the source). On the

² The leniency contract refers to an implicit agreement between majority and minority within a group in which majority listens to the minority's dissent voice to maintain the viability and cohesion of the group, and the minority also agrees that a focal change is unlikely

other hand, if the focal attitude is the minority attitude in the agent's local ingroup, then the agent examines consistency among the focal and related attitudes that the source has. At this time, the agent does not and/or cannot examine all the attitudes that the source has in its belief vector due to cognitive limitations.

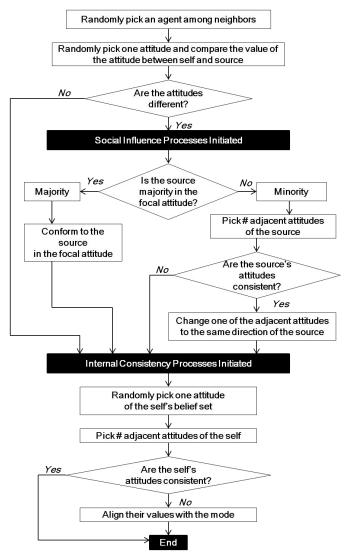


Figure 1. An Algorithm of Majority and Minority Influence and Internal Consistency Processes

As demonstrated by research on cognitive bias and bounded rationality (Simon 1957). For example, we are often *cognitive misers* (Fiske and Taylor, 1984, 1991) who prefer to process information efficiently by utilizing heuristics in our adaptive toolbox (Gigerenzer and Selten 2001, pp.37-50); people also act as *lay scientists* who systematically build knowledge from bits of evidence. Thus, in this model agents randomly select a few closely related attitudes from the source's belief vector and examine their consistency. If the closely related attitudes of the source are consistent (e.g., $a_{i,l}^k$

 $=a_{i+1}^k=a_{i+1}^k$ for focal attitude a_{i}^k)³, the agent changes one mismatched attitude out of the related attitudes (e.g., a_{i-1}^j or a_{i+1}^j) to match the source's focal attitude. However, if the attitudes are not consistent, the agent does not change any of its own attitudes. In sum, majority influence through conformity is narrow and immediate produces changes to the focal attitude while minority influence (through conversion) is less common and indirect, but has a broad impact when it happens.

Upon completion of the social influence process, the internal consistency process is initiated: these two processes are independent in the current model. In this stage, all agents turn their attention to their own belief set in order to tune their attitudes to be consistent. Each agent j randomly picks one attitude (i.e., a_h^j) in their belief set and examines if the chosen attitude and its adjacent attitudes have the same value. If the examined attitudes are not consistent, the agent tunes the chosen attitude toward the adjacent attitudes. This can be achieved, for example, by setting it to the modal value among adjacent values (if one exists) or by setting it to match a random adjacent attribute.

An Example

To demonstrate these processes we first consider a case in which each agent has five attitudes (M=5) with binary attitude values (i.e., $a^{j} \in \{0,1\}$). Agent j (marked in bold with gray shading in Figure 2) has an attitude vector of $a^{j} = \{11001\}$. Following our algorithm above, the agent considers its eight Moore neighbors (solid border) as its ingroup and makes a uniformly random selection of one to be its source for this turn. Assume the top left agent of the ingroup is chosen as k so $a^k = \{01111\}$. Next the agent randomly picks one attitude out of the five. Let's say a_3 , the 3rd attitude, is chosen. Agent *j* compares its value $(a^j{}_3=0)$ with the corresponding attitude of the source agent k $(a^k{}_3=1)$. Because $a^j{}_3\neq a^k{}_3$, agent j next evaluates the majority/minority status of the source agent k in the local ingroup in terms of the 3^{rd} attitude. Within a^{j} 's ingroup, there are six agents with $a_3=0$ and three agents with a=1: therefore the source's attribute value of 1 is in the local minority. Note that this evaluation is independent of the global prevalence of values for a_3 .

01001	10001	10101	01011	10101	00101	11001
00100	01001	11011	10000	11111	11001	01111
01110	01100	01111	10010	11011	10000	00010
01010	01110	11010	11001	01001	01101	10110
11011	10010	00001	10100	00110	10001	10111
10110	10101	01011	00101	10111	01111	00000
01011	01111	11101	00011	00110	11010	00100

Figure 2. An example demonstrating the agent updating process.

Now agent j evaluates whether the source k is consistent in its attitudes within a cognitive constellation of a^k_3 . The adjacent attitudes a^k_2 and a^k_4 are considered, so the source's attitudes are consistent if and only if a^k_2 , a^k_3 , and a^k_4 are all

³ The attitude vector is assumed to wrap around so that a_I^k and a_M^k are adjacent.

same. In this example, the values of a_2^k , a_3^k , and a_4^k are all 1, therefore the agent j changes one of its mismatched non-focal attitudes to match the source's value. Among the two constellation attitudes a_4^j is the only mismatched one, thus agent j changes the value of a_4^j to the source's value and so becomes $a_4^j = \{11011\}$.

However, if the agent from the bottom right of the ingroup (with attitude vector $\{00110\}$) is chosen as the source instead, due to its attitudinal inconsistency (i.e., $a_2^k = 0$, $a_3^k = a_4^k = 1$), agent j will not change its attitude at all. Or if the agent to the left of agent j is chosen as the source, agent j will not change its attitude because the source affirms agent j's attitude.

Going back to the first case, upon completion of the social influence process in which agent j has updated its belief vector to $a^j = \{11011\}$, the agent j next randomly picks one attitude out of five again...let's say a_2 . Agent j examines if the chosen attitude a^j_2 and its adjacent attitudes a^j_1 and a^j_3 have the same value (internal consistency). Because the examined attitudes are not consistent (i.e., $a^j_1 = a^j_2 = 1$, $a^j_3 = 0$), the agent tunes them to be the same value of the mode (in this case 1). As a result agent j's attitude vector becomes $a^j = \{11111\}$. Each agent undergoes the same process (either synchronously or asynchronously) for each iteration of the model.

Implementation in Netlogo with Minimal Conditions

We implemented this attitude updating algorithm in a cellular automata-style agent based model using Netlogo 5.0.5. To keep the dynamics clear we utilize the minimal conditions necessary to capture the effects of majority and minority influence and internal consistency. In our basic setup 1600 agents populate a 40-by-40 toroidal square-grid topology (no empty spaces). Ingroups are defined as the eight agents in the Moore neighborhood – just as above.⁴

Each agent has two attitudes and each attitude takes binary values: one attitude is represented as color $(a^j{}_i: 0 = \text{yellow}, 1 = \text{blue})$ and the other as shape $(a^j{}_2: 0 = \text{circle}, 1 = \text{square})$. Thus, there are four different agent states: yellow circle (i.e., $a^j = \{0, 0\}$), yellow square (i.e., $a^j = \{0, 1\}$), blue circle (i.e., $a^j = \{1, 0\}$), and blue square (i.e., $a^j = \{1, 1\}$). Consistent agent states are therefore the yellow circle and blue square. When checking consistency the agents simply change the focal attitude to match the other attitude.

We define a *cultural group* as a collection of contiguous (including corner connections) agents sharing the same value for all (both) attitudes. We are interested in the number of culture groups as a measure of the social diversity of a given set of attitudes. However, not all groups should necessarily be weighted equally: the smaller a cultural group is, the less it contributes to the society's diversity. We account for this by reporting both the number of groups and the effective number

of groups. The effective number of groups is calculated using the inverse Simpson index (Page 2010).⁶

Simulations and Results

We simulated the agent based model to answer two questions of interests: (1) Under which rules can an attitudinally diverse population persist? (2) Under which rules can social change occur (i.e., a nascent idea becomes prevalent)? For each combination of rules and initial conditions (described below) we performed 100 runs of the model until (a) all the agents have the same attitude, (b) no agents can change attitudes, or (c) 10,000 time steps (to terminate non-equilibrium runs and facilitate analysis of extremely long convergence times).

At each step we calculate the percent of the population of each color and each shape – since there are only two values for each attribute, percent yellow is one minus percent blue and it suffices to present percent blue and percent square. We also determine the number of groups of each attitude, combined attitude cultural groups, and the effective number of groups.

When do diverse cultures persist?

To identify conditions under which a diverse cultural population persists we initially create a population with equal numbers of both colors and both shapes. The two attitudinal dimensions are assigned independently and the spatial distribution of attitudes is uniformly random.

Majority influence. The dynamics of local strict majority influence processes are already well-known to science. These systems converge quickly (~50-100 iterations) into amorphous global patterns of each cultural group. The attitude distribution becomes skewed in a way that is dependent on initial conditions and stochastic elements, but with no systematic bias toward any attitude. When internal consistency processes are in play, majority influence performs much the same as if there were just one attitude with two values.

Minority influence. This process acts to balance the prevalence of each attitude, so with equal initial quantities and a random initial arrangement, minority influence alone is a continuous and balanced stochastic mixing of the attributes across the agents. Although there are typically hundreds of cultural groups, none of them are persistent. In order to maintain internal consistency the attitudes have to line up in alternating rows or columns thus creating 40 groups of equal size in equilibrium. However, it often takes many thousands of iterations to reach this equilibrium and in the process there are usually many fewer groups in play.

⁴ The ingroup for the majority/minority influence algorithm can be defined by a variety of neighborhood topologies (e.g. grids of different types or irregular networks), the geographies of moving agents, dynamic social connections, abstract associations, etc.

⁵ Squares are shaded slightly darker to be more easily discernable.

⁶ The inverse Simpson index is a measure of the effective number of types in a population. It is calculated as the inverse of the sum of the squared proportional sizes of the groups: $(\sum_i g_i^2)^{-1}$.

⁷ Obviously this value depends only on the dimensions of the space – we find 40 groups because we use a 40x40 world and the attitudes alternate. Using an odd number of spaces (e.g. 39x39) forces a vertical or horizontal strip of constant, stochastically balanced fluctuation.

Majority and minority influence without consistency. These processes together form large, consistent cultural groups with mixed-attitude boundaries that converge slowly (over thousands of iterations) to a single consistent attitude culture (Figure 3). The groups are consistent despite internal consistency being off because in minority influence, imitation only occurs when the targets' attitudes are consistent.

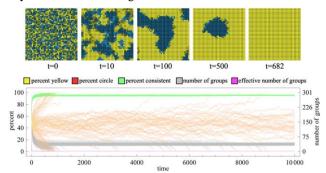


Figure 3: Results of simulation with majority and minority influence only. Screenshots from one run illustrate the spatiotemporal dynamics of consistent group formation and mixed boundaries. Plots for 100 runs showing that both outcomes are common. Most runs reach global conformity in <1000 iterations, but several take more than the 10,000 simulated. Lines appear orange because they are yellow layered on red when consistency is high.

Majority and minority influence with internal consistency. Majority and minority influences, in combination with internal consistency, create a system which has more, smaller cultural groups that come in an out of existence at a high rate. Even thought the system eventually converges, it typically takes much longer (ten times the iterations). Also, the majority and minority status of attitudes is erratic at the global scale: social change often happens, but it often switches back throughout the runs (Figure 4).

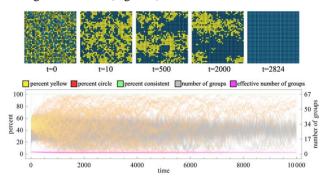


Figure 4. Results with majority and minority influence and internal consistency. Screenshots reveal the more fragmented nature of the groups and the lack of long-lasting coherent groups. Erratic time series and highly variable halting times across 100 runs reflect the stochastically balanced mechanisms.

Summary. Populations of diverse cultures will persist under either minority or majority influence alone (with or without internal consistency). When minority and majority influences work together, the system will always converge to a uniform population; however, this may take an extremely long time and even longer when internal consistency is in play. Interim

patterns of heterogeneity are more dynamic when internal consistency is in play.

Which mechanisms generate social change?

Next, we simulated the model to validate Crano's conjecture that minority influence at the local level can lead to social change at the global level via an internal consistency process. We also explore whether social change can be led solely by minority influence, and whether it is robust against the effects of majority influence. We initialize the system with a randomly chosen 1% of the agents holding the nascent attitude (yellow; i.e., $d_I = 0$) while maintaining equal initial populations of shape.

Minority influence alone. The indirect minority influence process will always spread a nascent attitude from 1% of the population toward 50% of a population. After reaching approximately half of the population, however, minority influence balances the two attitudes in a constant stochastic churning of both color and shape (as described above). Thus indirect minority influence alone cannot lead to social change. (Figure 5). Minority influence plus internal consistency also brings the population up to the roughly 50% point in every run, after which the dynamics mirror the case above.

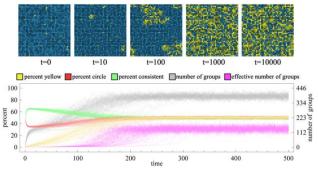


Figure 5. Screenshots show the growth in popularity of the nascent attitude up until the system becomes stochastically mixed at roughly equal proportions. Note also the alignment of the shape attitude in the early stages into locally balanced configurations. Though every run is distinct in its particular spatiotemporal arrangements, the dynamics are consistent across 100 runs. This plot is truncated to 500 iterations to highlight details of earlier periods.

Minority influence facing majority influence. In the face of majority influence, however, minority influence cannot spread the nascent attitude. Majority influence dominates minority influence process on the color attitude and so the system quickly converges to all blue. Thus social change does not occur. Minority influence (which imitates an attitude only when consistent) quickly eliminates circles from the population as well to establish a monoculture (Figure 6).

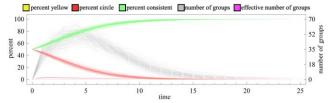


Figure 6. With only 1% of the population initially holding the nascent attitude, majority influence quickly dominates the effects of minority influence and uniformly spreads the prevailing idea.

Minority influence and internal consistency facing majority influence. Consistent with Crano's conjecture, the indirect minority influence process in combination with internal consistency *can* spread an initial minority position in the face of majority influence. Even though a system converges to the prevailing attitude in 75% of the runs (Figure8), there are chances for a nascent attitude to temporarily or even permanently be globally adopted (i.e., social change occurs) (Figure7).

Note that the initial 1% of agents that are yellow is immediately expanded in the first iteration when agents enact the self-consistency rule. Approximately half of the blue circle agents will make themselves consistent by becoming yellow circles – that's an additional 24.25% of the agents on average. We address this in future work (described below) by identifying the relationship between the number of initially consistent yellow agents and the success of social change.

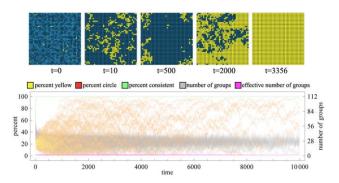


Figure 7. Screenshots from one run demonstrate the high volatility of the attitude proportions. The time series from 100 runs highlight this long-lasting volatile behavior.

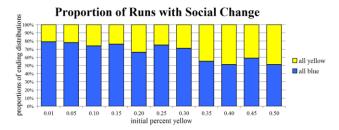


Figure 8. This chart shows the proportion of 100 runs for each ending distribution of the color attitude with all three rules activated. These runs were all taken to equilibrium regardless of the time step. Despite the high volatility of the dynamics, there is an increasing tendency to achieve social change with increasing initial percent holding the nascent idea.

Summary. In the face of the direct majority influence, social change can occur via the indirect minority influence process in combination with the internal consistency process. The indirect minority influence process alone (without the internal consistency rule) cannot spread an initial minority position

against the direct majority influence. However, this may be the result of an artifact of increasing the initial population of yellow agents through the internal consistency rule.

Discussion

The present study aims to investigate how direct majority and indirect minority influences at the local level, in combination with an internal consistency process, lead to cultural group formation and social change at a global level. The algorithm was constructed based primarily on the psychological rules posited in Crano's CCLC theory.

Majority influence immediately influences an individual to adopt the focal issue attitude due that is most common among its peers. The rules of minority influence are more complex. Ideas held by a minority of one's ingroup can also be influential, but they act indirectly on a non-focal attitude in the same attitude constellation. Furthermore, this influence only occurs when those holding the minority view are consistent in their own views. As indirect attitude changes accumulate, delayed focal change can occur through a motivation to maintain internal consistency. Indirect changes followed by consistency changes can lead an idea originally held by just a few individuals to take hold in a whole population in the right conditions. This is social change.

We tested for social change by seeding a population of 1600 agents randomly with 1% having the yellow attribute, while the shape attribute remains equally distributed. Majority influence alone, or with internal consistency, quickly removes the nascent idea form the population. Minority influence – alone or with internal consistency – reliably brings the nascent idea from 1% to 50% of the population, but naturally cannot achieve social change because the mechanism always promotes the attitude that is less popular.

When majority and minority influence are used together, but without internal consistency, majority influence dominates and the population quickly converges to the prevailing idea. When internal consistency is added, half of the initial bluecircle agents immediately become yellow-circle agents. This boost to the nascent idea holders often suffices to stop quick majority domination. Furthermore, the requirement that agents be consistent ensures that minority influence is engaged in each individual's mechanism because the minority attitude is always consistent. Although social change is possible under these conditions, it is unlikely. The likelihood depends on the initial percent of the population holding the nascent idea, and also on the micro-structure of their locations and the many stochastic elements in the model. Finally, although these simulations always end in one homogeneous equilibrium or the other, the time to reach the equilibrium can be extremely long (e.g., as much as 20,000 iterations). With mixed ideas persisting for so long, other mechanisms may interject to alter the balance.

In addition to social change, we are interested in the ability of these psychological mechanisms to maintain a diverse set of attitudes within a population. Cultural groups are communities of connected individuals who share all their (relevant) attitudes. The formation of coherent cultural groups from a random initial distribution of attributes is

important because these cultural groups can provide safe proving grounds for incubating new ideas.

Majority influence alone, or with internal consistency, succeeds in producing large cultural groups. These grouped patterns are equilibria because each agent's local neighborhood of eight agents has at least four agents of the same attitude. Without consistency checks the attributes are influenced independently, creating a multi-layered pattern in which all combinations of the attitudes persist.

In contrast to the equilibrium cultural groups formed by majority influence, the groups formed by minority influence are in constant flux. Without internal consistency, the process acts as a balancing stochastic process that ensures all attributes are roughly equally represented. There are a large number of cultural groups, but they are very small and are not persistent. They fail to capture the phenomena we wish to pick up as cultural groups; instead, they are just temporary patches of attributes. Adding consistency induces another equilibrium condition, but a mixed one. Every agent must be the minority of it local neighborhood and this is only achievable by having an even number of straight rows or columns. This is also not the sort of cultural groups identified in the literature, but rather an artifact of the model.

Combining all three mechanisms produces an interesting dynamic of cultural groups reminiscent of coherent structures in Conway's Game of Life (Conway 1970), although the stochastic elements in the current model complicates a microscopic analysis of recurrent patterns. Large and small groups of both consistent cultures can persist for long periods, "float" across the screen, rise up, and die out. As is the case with any cellular automata, these seeming movements are the result of changing attributes of stationary agents, but the population-level patterns are intriguing dynamical features of these mechanisms not anticipated by the social psychological theories.

Other features of these simulations worthy of discussion include the long time-spans of many runs, the non-equilibrium nature of some behaviors, and the volatility of populationlevel patterns. The preferred analytical technique among many social scientists involves equation-based modeling that focuses on equilibrium of outcomes or distributions. However, our results highlight the need to model and understand the dynamics of the long time-spans before equilibrium conditions are reached. Furthermore, in some cases we can foretell the outcome after a few iterations through feedback effects among clusters. In other cases (such as when all three mechanisms are active), the population can shift widely from one extreme to the other in just a dozen iterations, making reliable prediction impossible. Although we report the distribution of equilibria as a function of initial population attributes, a deeper analysis of the interim dynamics is more likely to improve our understand of the phenomena.

Limitations and Future Direction

Our goal in this paper is to present a minimal model capable of embodying the mechanisms of majority and minority influence as described in the social psychological literature. These theories, however, are purported to have much wider and deeper application. In order to fully engage the literature expansions must be made to both the simulations and the theories.

Expanded attitude space. The first obvious augmentation is to incorporate longer attitude vectors and more variations per attitude. We would like to test the sensitivity of the dynamics to the size of this attribute space a la Axelrod (1997), but the theories themselves need further refinement to handle influence in cases in which no attribute holds a majority or clear minority. What influences reign in a pluralistic environment?

More sophisticated agents. Our agents fully populate a torus space in a regular lattice – a common feature of theoretical agent-based models. More realistically agents would migrate, change interaction partners, have heterogeneous neighborhoods, and these behaviors would also be contingent on local opinions. Furthermore they could reproduce, mutate, or undergo selection processes. With this prototype model, we are able to clearly ascertain the link between agents' behavioral rules and system-level patterns.

Internal consistency force. Our findings show that the internal consistency process plays a key role in social change. The next question is how the strength of internal consistency affects social change; specifically, what is the relationship between the strength of internal consistency checks and the occurrence of social change? This question can be answered by varying the probability that a check is performed, and/or by changing the number of attitudes that are put in line (after we add more attributes).

Objective vs. subjective issues. Attitudes may address objective or subjective issues. Objective issues have a correct belief, such as the weight of a particular ox. Subjective issues (aka, mere opinions such as what color is "in" this season) reflect the converging of ideas in the population. Crano argues that for objective issues, a nascent attitude is scrutinized by counter-attitudinal individuals, which may lead to an immediate focal change if the attitude is correct. Depending on objectivity/subjectivity of issues, different rules may govern agents' influence processes. In future studies, we can construct and compare two models where different rules are implemented depending on objectivity/subjectivity of issues.

Conclusions

Majority and minority influence are important and well-studied topics in the field of social psychology. Due to the limitations of empirical methodologies, however, theories and empirical research focus mainly on explicating interpersonal and intra-individual psychological rules of social influence. They neglect how such rules lead to larger level group phenomena such as social change and cultural group formation. Most of the claims linking interpersonal/intra-individual-level rules and group/society-level patterns are merely conjectures. The present study is an attempt to provide a detailed connection between the individual and social layers through simulation.

We implement the social influence rules postulated in Crano's CCLC theory in an agent-based model with minimal assumptions. Simulation results demonstrate that in the current formulation the indirect minority influence alone cannot lead to social change in the face of the direct majority influence. However, our results also show that in combination with an internal consistency rule, the three processes together can in some scenarios lead to social change. Furthermore, both the majority and minority influence processes can lead to the formation of diverse cultural groups of distinct kinds.

The current study shows the partial validity of the conjecture that the social change at a societal level occurs through indirect minority influences process at the local level in combination with the internal consistency process. Our study reveals that all three forces are necessary for social change. The indirect influence process triggered by counterattitudinal ingroup minority alone cannot make social change occur in the face of majority domination. What is necessary is individuals' motivation to maintain internal consistency in their attitudes, to emulate consistent minority attitudes, and to adopt popular ideas both old and new.

Acknowledgements

We would like to gratefully thank Dr. Scott Page for his feedback on the model in the early stage of this project and Dr. William Crano and Dr. Patrick Grim for their insights for future directions.

References

- Alvaro, E. M., & Crano, W. D. (1997). Indirect minority influence: Evidence for leniency in source evaluation and counterargumentation. *Journal of Personality and Social Psychology*, 72, 949-965.
- Asch, S. E. (1951). Effects of group pressure on the modification and distortion of judgments. In H. Guetzkow (Ed.), *Groups, leadership* and men (pp. 177-190). Pittsburgh, PA: Carnegie Press.
- Asch, S. E. (1952). *Social Psychology*. Englewood Cliffs, NJ:: Prentice Hal
- Asch, S.E. (1955). Opinions and social pressure. Scientific American, 193, 35–35.
- Axelrod, R. (1997). The dissemination of culture a model with local convergence and global polarization. *Journal of conflict resolution*, 41(2), 203-226.
- Clark, R. D., III (1990). Minority influence: The role of argument refutation of the majority position and social support for the minority. European Journal of Social Psychology, 20, 489–497.
- Conway, J. (1970). The game of life. Scientific American, 223(4), 4.
- Crano, W. D. (2010). Majority and minority influence in attitude formation and change: Context/categorization – leniency contract theory. In R. Martin & M. Hewstone (Eds.), *Minority influence and innovation: Antecedents, processes, and consequences* (pp. 53-77). New York: Psychology Press.
- Crano, W. D. (2000). Social Influence: Effects of leniency on majorityand minority-induced focal and indirect attitude change. Revue Internationale de Psychologie Sociale, 15, 89-121.
- Crano, W. D., & Alvaro, E.M. (1998). Indirect minority influence: The leniency contract revisited. Group Process and Intergroup Relations, 1, 99-115.
- Crano, W. D., & Chen, X. (1998). The leniency contract and persistence of majority and minority influence. *Journal of Personality and Social Psychology*, 74, 1437-1450.
- Crano, W. D., Seyranian, V. (2009). How minorities prevail: The context/comparison-leniency contract model. *Journal of Social Issues. Vol.* 65, pp. 335-363.
- Crutchfield, R.S. (1955) Conformity and character, *American Psychologist 10*: 191-8.

- Festinger, L. (1954). A theory of social comparison processes. *Human Relations*, 7, 117 140.
- Festinger, L. (1957). A Theory of Cognitive Dissonance. Stanford, CA: Stanford University Press.
- Fink, E. L., & Kaplowitz, S. A. (1993). Oscillation in beliefs and cognitive networks. In W. D. Richards Jr. & G. A. Barnett (Eds.), *Progress in communication sciences* (Vol. 12, pp. 247-272). Norwood, NJ: Ablex.
- Fiske, S.T., & Taylor, S.E. (1984/1991). Social Cognition (2nd ed.). New York: McGraw-Hill.
- Gigerenzer, G., & Selten, R. (2001). Bounded rationality: the adaptive toolbox. Cambridge, Mass, MIT Press.
- Judd, C. M., Drake, R. A., Downing, J.W., & Krosnick, J. A. (1991).
 Some dynamic properties of attitude structures: Context-induced response facilitation and polarization. *Journal of Personality and Social Psychology*, 60, 193 202.
- Martin, R., & Hewstone, M. (2001). Determinants and consequences of cognitive processes in majority and minority influence. In J. P. Forgas & K. D. Williams (Eds.), Social influence: Direct and indirect processes. (pp. 315-330). New York, NY US: Psychology Press.
- Martin, R. & Hewstone, M. (Eds.) (2010). Minority influence and innovation: Antecedents, processes and consequences. Hove, E. Sussex: Psychology Press.
- Martin, P. Y., & Martin, R. (2006). The effects of caffeine consumption on direct and indirect majority and minority influence. *Journal of Applied Social Psychology*, 36, 1961–1979.
- McGuire, W. J. (1990). *Dynamic operations of thought systems*. American Psychologist, 45(4), 504-512.
- McGuire, W. J., & McGuire, C. V. (1991). The content, structure, and operation of thought systems. In R. S.Wyer & T. Srull (Eds.), *Advances in social cognition* (Vol. 4, pp. 1 78). Hillsdale, NJ: Erlbaum.
- Moscovici, S. (1976). Social influence and social change. New York: Academic Press.
- Moscovici, S. (1980). Toward a theory of conversion behavior. In L. Berkowitz (Ed.), *Advances in experimental social psychology* (Vol. 13, pp. 209 239). New York: Academic Press.
- Moscovici, S. (1985). Innovation and minority influence. In S. Moscovici, G. Mugny, & E. Van Avermaet (Eds.), *Perspectives on minority influence* (pp. 9 52). Cambridge, UK: Cambridge University Press.
- Moscovici, S. & Faucheux, C. (1972), Social influence, conformity bias, and the study of active minorities, in BERKOWITZ, L. (ed.), *Advances in Experimental Social Psychology*, New York and London, Academic Press, Vol. 6, pp. 149-202.
- Moscovici, S., Lage, E., & Naffrechoux, M. (1969). Influence of a consistent minority on the responses of a majority in a color perception task. *Sociometry*, 32, 365–379.
- Mucchi-Faina, A., Pacilli, M.G., & Pagliaro, S. (2010). Minority influence, social change, and social stability. Social and Personality Psychology Compass, 4(11), 1111-1123.
- Newcomb, T.M. (1943). Personality and Social Change: Attitude Formation in a Student Community. New York: Dryden Press.
- Page, S. (2010). Diversity and Complexity. Princeton University Press.
- Sherif, M. (1936). *The Psychology of Social Norms*. New York: Harper Collins.
- Sherman, D. K., & Cohen, G. L. (2006). The psychology of self-defense: Self-affirmation theory. In M. P. Zanna (Ed.), Advances in Experimental Social Psychology (Vol. 38, pp. 183-242). San Diego, CA: Academic Press.
- Simon, H. A. (1957). Models of man, social and rational: Mathematical essays on rational human behavior in a social setting. New York, NY: John Wiley & Sons.
- Steele, C. M. (1988). The psychology of self-affirmation: Sustaining the integrity of the self. In L. Berkowitz (Ed.), Advances in experimental social psychology (Vol. 21, pp. 261-302). New York: Academic Press.
- Wood, W., Lundgren, S., Ouellette, J. A., Busceme, S., & Blackstone, T. (1994). Minority influence: A meta-analytic review of social influence processes. *Psychological Bulletin*, 115, 323 –345.

The Origin of Culture: Selective conditions for Horizontal Information Transfer

Miguel Gonzalez¹, Richard Watson¹, Jason Noble¹ and Seth Bullock¹

¹Institute for Complex Systems Simulations, University of Southampton, UK SO17 1BJ mgc1g11@soton.ac.uk

Abstract

Culture is a central component in the study of numerous disciplines in social science and biology. Nevertheless, a consensus on what it is and how we can represent it in a meaningful and useful way has been hard to reach, especially due to the multifaceted aspects of its nature. In this work we dissect culture into its most basic components and propose horizontal information transfer as the most crucial aspect of it. We discuss the two fundamental processes that are required for culture to emerge in an evolutionary context, namely: high imitation error rates and survival selection. To show how each of these components affect the emergence of culture, a genetic algorithm was explored for a range of conditions. Here, we formalize when and how a population is said to move from biological to cultural evolution and why such a transition radically changes its evolutionary dynamics. Our results suggest that horizontal transfer of information in cultural systems requires the evolution of survival enhancing traits rather reproduction enhancing ones. We consider this requirement to be key for the evolution of rich cultural systems, like the one present in humans.

Introduction

Not only has the human species been able to adapt to a massive range of environments, but we have also transformed them to suit our needs. Culture is said to be one of the primary drivers for the accelerated pace at which we are able to establish and grow in new areas. (Boyd, 1985; Richerson and Boyd, 2006). Its origins are not fully understood but we know that our biological evolution must have set the stage for its emergence. In other words, just before we became a cultural species, we evolved to a point where primitive forms of social learning were possible (Richerson and Boyd, 2006). Literature offers numerous concepts of culture, for some researchers in the social sciences a superorganic view of culture, that is detached form biology, seems to offer a better and more useful explanation for the phenomena that we observe in our societies (Kroeber, 1948; Ingold, 1986). From this point of view culture would start from our biological nature but once it takes off it will be an independent process with no major feedbacks with our biological traits. For others, biology is in constant relation with culture in deeply intertwined ways (Richerson and Boyd., 2001; Rogers, 1988). We agree, as many studies suggest, that culture has been shaped by our biological history and in return our genetic traits have evolved in response to it (Richerson and Boyd, 2006; Holden and Mace, 1997). Nevertheless, in this work we focus on the disengagement process between genes and phenotype for a particular set of evolving characteristics represented as a vector of binary traits to be optimised. This might associate our model with a more superorganic view of culture, but as it will be discussed, our analysis focuses on the evolution of traits that would spark the emergence of culture rather than on the long term interactions between genes and cultural evolution.

We maintain that culture is an outcome of genes giving rise to and interacting with an evolving environment of ideas and behaviours. Cultural variants, the equivalent of alleles in genetic systems, compete for a space in our minds and are transmitted by means of social learning (imitation and teaching), they can alter our behaviour and in many ways override some of our most basic hardwired instincts. But culture, is also population level phenomena and needs to be defined as one. Considering this, we could say that culture is: the set of behavioural traits that are not the direct result of genetic expression but rather the product of an evolving pool of variants that are stored and transmitted within and between overlapping generations of individuals, by means of social learning. This important concept is discussed in the literature and sets a starting point for our dissection (Avital and Jablonka, 2005; Richerson and Boyd, 2006). A way to understand such a concept is to think of an individual developing in social isolation. This individual would not have the culturally evolved traits that form the phenotype of its socially interacting peers, but not all the traits missing in a socially isolated organism can be considered culture. Some are the result of instinctive interactions between members of a social group and these are also hard wired in genes. It is only when we take these life history and socially instinctive traits out, that we can identify culture as the missing set of traits.

As we said before definitions of culture are numerous and,

depending on which one we select, it can be identified as a unique trait of our species or a repeatedly occurring one in nature (Laland and Janik, 2006; Laland and Hoppitt, 2003; Avital and Jablonka, 2005; Heyes and Galef, 1996). Our definition includes several animal cases like species of birds that learn their song by imitation (Jenkins, 1978; Heyes, 1994), chimpanzees that learn to use tools and simple protocols for nut cracking (McGrew, 1998), cetaceans imitating hunting strategies and mating calls (Rendell and Whitehead, 2001) and others (Laland and Galef, 2009). Even though in animal cases the complexity and repertoire of culturally evolved traits is limited, it is in principle sensible to consider a general mechanism for the emergence of culture. Our intention, as well as previous models on the evolution of culture, is to show the adaptive character of it. But before we consider the adaptive value of culture we need to describe its peculiarity and some valuable existing contributions.

At the population level culture acts as a whole new evolutionary system. In this system, evolution can only take place as long as all the ingredients of an evolutionary process are present: Reproduction, Inheritance, Mutation and Selection. The equivalent to reproduction and inheritance is implicit in the act of social learning: information gets passed from one individual to another and most of it is received. Mutation is then introduced by errors in imitation; these errors can have positive or negative effects and in this sense are fundamentally different from the effect of individual learning, which tends to improve or adapt variants during lifetime. The balance between inheritance and mutation is crucial for any evolutionary process to occur; too much mutation and the system fails in an error catastrophe. On the other hand, excessive fidelity on information transfer and the system gets stuck on a single solution (Jong, 2002). Some critics of cultural evolution point out that the equivalency of genes with cultural variants (AKA memes) is not a sensible one, due to the low signal to noise ratio of cultural transmission (Burman, 2012). This is a valid observation, but we seem to intuitively understand that ideas learned from others are very similar to the original ideas that those others hold; if this was not the case communication would be impossible. A way to reconcile these positions is to consider the convergent nature of learning towards useful forms of variants (Dawkins, 1976). When we collate new information, noise can be dramatically reduced with further practice or reinforcement because we tend to converge to the useful form of that variant. An example: nut cracking techniques that might be transmitted with a lot of noise from adults to young chimpanzees in natural populations. Maybe the original attempt is different from the proper technique but this will eventually converge to a copy of the original strategy due to the useful result obtained by it (i.e., the nut!) .

Selection in cultural systems can come in a variety of forms, different biases in transmission have been identified and some of them are directly related to fitness enhancement (Enquist et al., 2007) others are frequency dependent (conformism) (Henrich and Boyd, 1998) and some are driven by social status (Richerson and Boyd, 2006). These biases have been discussed in the literature as potential explanations for the adaptive nature of culture. Most authors refer to the seminal paper of Rogers (1988), in this paper a very simple model showed how culture is not inherently adaptive just by means of its defined characteristics or the characteristics of an evolutionary process under natural selection (Rogers, 1988). Its premise is that adapting to a new environment by means of finding an individual solution implies a cost for the learner; this cost can be avoided by imitating others, so that in a non-changing environment an invasion of imitators would be the ESS of the system due to the avoided cost. But if environmental change gets taken into consideration then individual learners increase in frequency due to the advantage that they have over imitators copying environmentally uncorrelated information. In this way, the ESS would be a mixture of individual learning and imitation determined by the rate of change in environmental conditions. Rogers's model found that the point of equilibrium does not confer any adaptive advantage to a cultural population vs. a noncultural one; the evolution of culture considering the lack of intrinsic adaptive value is known as the Rogers's Paradox.

To solve this paradox several interesting and useful theoretical models have been developed, including extensions with transmission biases (Enquist et al., 2007), spatial dynamics (Boyd and Richerson, 1988) and population structure (Rendell et al., 2010). Here, we would like to take a step in a different direction. Rather than finding equilibria of strategies, biases or spatial structure that increase the adaptive value of culture we focus on the exploratory nature of cultural systems when it comes to finding new solutions in a fitness landscape. Particularly, we focus on the hybrid system present at the moment of transition, showing how culture originates and which are the minimal conditions for it. Our model does not consider environmental change because it focuses on the exploration and exploitation properties of biological vs. cultural systems when challenged to find novel solutions to temporally stable problems. Also, the model does not include individual learning; it merely shows the transition from instinctive behavioural traits to culturally acquired ones.

We recognize the inherent complexities of gene-to-instinct mapping and the potentially intricate mechanism for overwriting such behaviours with imitation. However, we consider that for the scope of our question such processes can be drastically simplified. Is for this reason, that in the description of our model a simple 1-to-1 map of genes to behaviour is proposed and the potential to overwrite such information with a copied behaviour is controlled by a binary switch.

Before we move our attention to the model description let us focus on individual learning and vertical imitation, both of them important for the evolution of culture, but not included in the scope of our analysis.

There is no doubt that individual learning and cultural evolution are constantly interacting and affecting one another (Richerson and Boyd, 2006; Avital and Jablonka, 2005; Rogers, 1988). Nevertheless our question looks at the adaptive value of culture as an evolutionary algorithm and this allows us to treat individual learning as an extension of instinctive behaviours. Here we consider both to be, albeit in different ways, a result of genes and developmental conditions and even though they may differ at a mechanistic level the notion that individual learning has a cognitive tax is not relevant for the scope of our question.

Also, it is sensible to assume that vertical transfer of variants is the ancestral form of imitation. Parents tend to be the most present and readily available source of information to imitate (Avital and Jablonka, 2005). Nevertheless, systems that depend entirely on vertical imitation would not be fundamentally different from genetic systems and would merely represent the same algorithm with an added substrate for information transmission. In order for culture to represent a novel evolutionary system where ideas and not organisms are selected, horizontal transmission should be included. For this reason we concentrate mostly on this sort of transmission in our analysis, only exploring its vertical counterpart in a later section. From here on we will equate the evolution of culture with the evolution of horizontal transmission, unless otherwise specified.

The Simulation Model

The model is a steady-state genetic algorithm in which each individual is represented by two strings of bits. The first string is considered the phenotype and the second the genotype. A 1-to-1 mapping from genotype to phenotype represents genetic expression. Fitness evaluations are made considering only the phenotypic information. Genotypes have an extra bit that acts has an imitation switch: if its value is 1 the individual will substitute the genetic expression string of bits in its phenotype for the phenotype of a randomly selected individual in the population. If the value is 0 the genotype string is copied into the phenotype, excluding the imitation bit. The action of imitation takes place at birth for each individual and once it has happened the phenotype remains unchanged for its lifetime.

Phenotypic imitation has an associated mutation rate, here described as μ_p . In the same way, genetic reproduction incorporates a mutation rate μ_g which also affects the extra imitation bit (switch) in the genotype string. Genetic expression, that is genotype to phenotype copying when the switch value is 0, does not include any errors. For all the results shown in this paper the value of μ_p and μ_g are fixed with a bitflip chance of 0.01 and 0.001 respectively, unless specified.

The population is initially set with an imitation switch

value of 0. and the second half of both strings of bits is set to 0s and, the first half is set to 1s. This is done to avoid mutation biases that would create an upward trend in fitness under lack of selective pressure. The fitness landscape explored in this paper is the one defined by the sum of ones in the string of bits. In this way a string of all-1 represents the optimal solution. For all the results shown here the length of the bit string is two hundred bits (L = 200), and the size of the population is one hundred individuals (N = 100). The general results can be reproduced with larger populations and larger strings of bits; smaller populations can produce different results between individual simulations but on average they would follow the behaviours here described. The processes we discuss here are relevant when the bit string is long enough for a search and optimization period of several iterations to take place. Single-digit bit strings, for example, might not reproduce the results we describe here.

Selection is established by the joint action of a reproduction function and a death function. Reproduction selects an individual of the population with a probability P_r from a Boltzmann-weighted function distribution of Fitness (Eq. 1). The death function uses the same method but with the complement of the number of ones in the phenotype vector rather than the fitness value $L-L_1$ calculating in this way the chance of dying as P_d (anti-fitness) (Eq. 2). On each iteration a mutated copy of the individual selected by the reproduction function will substitute the individual one selected by the death function.

$$P_r = e^{\frac{L_1}{x_r}} \tag{1}$$

$$P_d = e^{\frac{L - L_1}{x_d}} \tag{2}$$

Different combinations of strength between survival and fertility were considered by changing the value of the exponents x_d and x_r . Random selection for reproduction and random selection for death are also included in our results. Figure 1 illustrates the algorithm we have described here.

This model does not include crossover functions; we recognize that recombination advantages can be added to the cultural process by increasing the source of possible models to follow. Future work would be oriented towards exploring the advantage of cultural multi-parent crossover vs. bi-parental crossover in genetic reproduction. Here, we decided to focus on the effects of horizontal transfer of information.

Results

Simulations were run for a range of values (mutation rates, population size, vector length and selection strength). Results suggest that the relation between mutation rates and the type and strength of selection are critical for the evolution of horizontal information transfer to take place. Before

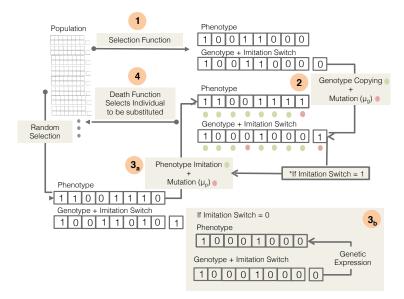


Figure 1: The diagram shows an iteration of the algorithm: (1) the selection function picks an individual from the population, (2) its genotype string is copied including the imitation switch (mutations may occur), (3a) if the imitation switch is equal to one, a random individual from the population will be selected and its phenotype will be copied with an associated imitation error. (3b) if the switch is equal to zero the bit string in the genotype will be copied into the phenotype with total fidelity. (4) The resulting combination of phenotype and genotype will replace an individual selected by the death function.

discussing these critical points it is important to describe the three possible behaviours emerging from these simulations. Figure 2 describes these scenarios.

In case A disengagement between phenotype and genotype fitness takes place early in the simulation along with a rapid growth in the proportion of imitators. Here, survival selection establishes a fitness-proportional life length for cultural variants. This case distinctively shows how the evolutionary process is taken over by horizontal imitation. Selection only evaluates phenotypes while genes get masked as soon as imitation frequency rises. Further evolution increases the gap between phenotype and genotype, making it even more costly to stop imitating and start expressing information from the genome. Eventually an evolutionary process entirely dependent on social learning finds the optimal solution and information is then maintained in individuals' actions rather than genes.

Case B is similar to A but here the selection strength for the phenotype is not enough to reach the optimal solution, this case occurs when survival selection is low $(x_d >> 1)$. Imitation, with its high mutation rate, will initially outper-

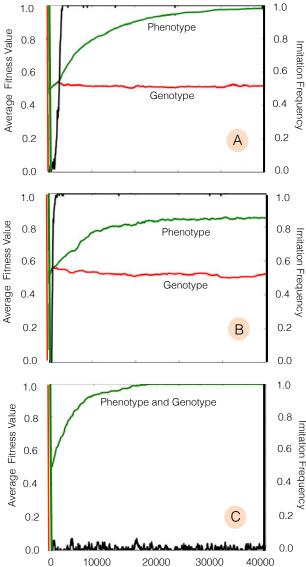


Figure 2: Single runs showing the three different scenarios: (A) Culture emerges early in the simulation under strong survival selection and the fitness disengagement between phenotype (green line) and genotype (red line) is accompanied by the sudden increase of imitation frequency (black line) ($x_d = 1$, $x_r = \infty$). (B) The fixation of imitators is also accompanied by fitness disengagement, but in this case the balance between selection and mutation prevents the phenotype form reaching the optima while the high frequency of imitation masks the genotype from selection. This case takes place under weak survival selection ($x_d = 5$, $x_r = 1$). (C) Disengagement is not present when survival selection is absent, both imitation rates stay low. Genetic evolution dominates ($x_d = \infty$, $x_r = 1$).

form the search ability of genes but eventually fail to reach the optima after disengagement. Selection on the genotype is masked making it impossible for genes to catch up. Variations of case B are rarely found under lack of survival selection; in these cases there is a chance that genetic fitness will eventually match phenotype fitness, if this trend goes beyond mid way the solution gradient (average of a drifting pattern), imitation frequency will drop to zero and the population will evolve genetically from then on.

Case C shows a standard genetic evolution scenario where both phenotype and genotype correlate all the way to the optimal solution; in this case imitation rate remains low. These genetic systems relay on vertical inheritance, which, under reproductive selection alone, is the only way to consistently optimize solutions. Simulations tend to converge to case C under lack of survival selection.

Types of Selection in Cultural Systems

In order to find the minimal conditions for horizontal transfer to evolve, we explored a range of combinations of the Boltzmann exponents in equations 1 and 2. These exponents control the strength of fitness based selection for survival P_d and reproduction P_r . The larger the value of x_d and x_r the weaker the strength of selection. Figure 3 shows a distribution of the three different cases discussed in the previous section for different combinations of Boltzmann exponents.

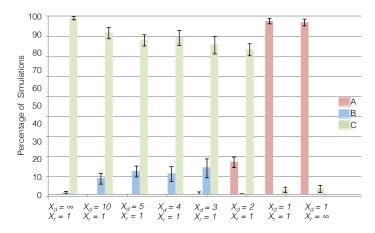


Figure 3: The percentage distribution of cases A, B and C is shown for different combinations of selection strength. Ten replicates of 100 individual simulations were run for each combination; positive and negative error bars represent a single standard deviation.

Cultural cases A and B are more frequent when the strength of survival selection increases but when selection only affects reproductive success case C is dominant. The reason for this has to do with the way horizontal imitation breaks the very notion of inheritance. Without inheritance,

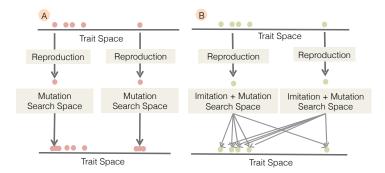


Figure 4: Comparative analysis of cultural vs. genetic search. (A) The genetic case explores an area restricted to the vicinity of the parental position in trait space. (B) Cultural evolution with horizontal information transfer establishes a common search area defined by the current distribution of phenotypes in the population, effectively getting rid of vertical inheritance and eliminating the impact of reproductive selection.

the number of offspring that an individual has is irrelevant.

In genetic reproduction each individual explores the trait space in independent lines from one another. In other words, the space of possibilities for a newborn is limited to the area where its parents currently are plus a surrounding space established by the genetic mutation rate. With horizontal imitation every single newborn in the population has the same search area, that is the current distribution of phenotypes in search space plus the imitation error area around them. A graphical representation of this is shown in Figure 4.

For cultural systems with horizontal transfer it does not matter who produces the new individuals because the algorithm stops looking at them as units of selection and instead it focuses on cultural variants. The bottom line is how long can you survive and serve as a model for others, rather than how many others can you produce. Our model links the selection of variants to the survival of individuals just like in a natural selection scenario, but it is important to reiterate, that this is not the same as selecting for the current state of independent search paths as it happens in biological evolution (figure 4A). This subtlety is a fundamental difference that changes the focus of selection from organism to variants.

These findings lead us to think that in order for horizontal transfer to evolve, with no transmission biases, a strong component of survival fitness should be present for a set of traits in the population. By extension, these types of traits would be fundamental for the emergence of culture.

In natural populations this claim is hard to test because behavioural traits have different mixtures of reproductive and survival fitness components; most species respond to some form of transmission bias (Avital and Jablonka, 2005) and evolutionary races to find new solutions for problems are hard to spot and trace. Nevertheless, we consider the findings of our model interesting from an A-Life perspective, especially due to its characteristic phenotype genotype disengagement behaviour and potential for extensions.

The Culture Advantage of Mutation Rates

Horizontal transfer without any transmission biases can generate faster adaptation compared to genetic evolution. This is mainly because the error rate of imitation is higher than mutation rates in genes. In figure 5 the average path for one hundred replicates compares the performance of a horizontal transfer cultural system (Cultural HT) with a vertical one (Cultural VT). In Cultural VT parents are the only models to imitate; this system is slightly different from a purely genetic case because the evolution of imitation frequency is still considered and both genes and variants are evolving. For comparison, a genetic case with mutation rate equal to the imitation error in cultural systems (Genetic HM) and a genetic system with a low mutation rate (Genetic LM), are included.

The trajectory described by Genetic LM is similar to that of a horizontal transfer system where the imitation error is set to the same low value as the mutation rate of genes. In similar fashion, if both mutation and imitation errors are set to a high value, the trajectory described by Genetic HM will take place. This shows how the advantage of cultural systems in Cultural VT and Cultural HT relies heavily on high error rates in imitation when compared to its low mutating genetic counterpart.

In nature, genetic mutation rates have a baseline and its random occurrence is considered a physicochemical constraint of DNA replication. Nevertheless, there are corrective mechanisms that can attenuate the effect of mutation during genetic replication (Lodish et al., 2004). The evolution of these mechanisms is considered an adaptation to the high level of contingency existing in development. Higher mutation rates could easily evolve in an organism but such a trait is undesirable due to the associated frequency of deleterious mutations arising from it. In this way, it is sensible to say that genes cannot afford to mutate as much as cultural variants due to the collateral effects on existing adapted traits.

If culture, with its higher mutation rate, is to take over genetic evolution it has to find a way to shield finely tuned behaviours that are critical for life. For this reason we would like to make the point that the term cultural evolution as described in this work should be reserved for newly established problems which could be the result of a sudden environmental change or the introduction of a new social group in the community. Our model does not describe this mechanism in detail and it merely assumes that this is the sort of problem that the hybrid system needs to tackle.

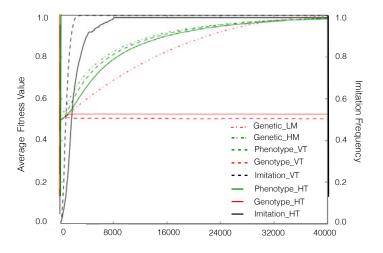


Figure 5: The average trajectory for a hundred individual simulations is shown for an horizontal transfer algorithm HT (continuous lines) and a vertical transfer algorithm VT (segmented line). Dot and segment lines represent paths for genetic evolution algorithms with mutation rates of 0.001 (Genetic LM) and 0.01 (Genetic HM). Here is shown that the fundamental advantage of cultural algorithms relies on the high error rate ($x_d = 1, x_r = \infty$).

Extensions

The model here presented is intended as a building block for future research endeavours. We think it is minimal in its assumptions and yet it produces the fundamental behaviours associated with culture. Current analysis on the embedded evolution of horizontal (vs. vertical) imitation is being developed. In addition, we are looking at the previously mentioned crossover advantage when having numerous models to copy from.

Transmission biases could easily be added by making the selection of a model to copy a function of its fitness, frequency or individual social status. Population structure is just a networked implementation away. On a networked model models to imitate would come from neighbour nodes rather than from the whole population; such a constraint could stimulate the evolution of culture by creating clusters of solutions that rarely exchange information, exploiting in this way the advantage of weak links in social networks (Csermely, 2000; Granovetter, 1983).

A more sophisticated approach would include the idea of frequency dependence and social games. Correlation between strategies in social games has been shown to increase the total benefit for individuals. An extension to the model here explained could show how culture can promote such correlations taking advantage of social learning and imitation in a game theoretic context.

Conclusions

Our model shows basic attributes of the evolution of culture. That we see the characteristic fitness disengagement between the phenotype and the genotype co-occurring with the fixation of imitation provides a clear fingerprint for culture. Within the constraints of our model, two conditions are necessary for horizontal transmission to evolve i) a high imitation error rate that provides an exploratory advantage and ii) a strong survival selection function that maintains selective pressure when imitation takes over. The mechanism here presented showed how genotypes can be masked from selection when imitators increase in frequency, and as a result a clear disengagement between phenotype and genotype occurs. We consider this key for the emergence of culture as defined in this work. Potential future extensions that could offer a clearer picture of the interaction between vertical and horizontal transmission, along with the possible effects of transmission biases, population structures, and social games are being developed.

We hope our findings lead the way to a general simulation framework to explore culture emergence and cultural evolution; the model is simple and stripped of most complexities obscuring the basic attributes of hybrid gene-culture systems, which makes it an excellent candidate for future development.

Acknowledgments

This work was supported by an EPSRC Doctoral Training Centre grant (EP/G03690X/1)

References

- Avital, E. and Jablonka, E. (2005). *Animal Traditions*. Cambridge University Press.
- Boyd, R. (1985). *Culture and the Evolutionary Process*. University of Chicago Press.
- Boyd, R. and Richerson, P. J. (1988). An evolutionary model of social learning: The effects of spatial and temporal variation. In Zentall, T. and Galef, B. G., editors, *Social Learning: Psychological and Biological Perspectives*. Lawrence Erlbaum Associates.
- Burman, J. T. (2012). The misunderstanding of memes: Biography of an unscientific object 1976-1999. *Perspectives on Science*, 12(1).
- Csermely, P. (2000). Weak Links: The Universal Key to the Stability of Networks and Complex Systems. Springer.
- Dawkins, R. (1976). The Selfish Gene. Oxford University Press.
- Enquist, M., Eriksson, K., and Ghirlanda, S. (2007). Critical social learning: A solution to rogers' paradox of nonadaptive culture. *American Anthropologist*, 109(4).
- Granovetter, M. (1983). The strength of weak ties: A network theory revisited. *Sociological Theory*.

- Henrich, J. and Boyd, R. (1998). The evolution of conformist transmission and the emergence of between-group differences. *Evolution and human Behaviour*.
- Heyes, C. (1994). Social learning in animals: Categories and mechanism. *Biological Reviews*.
- Heyes, C. M. and Galef, B. G., editors (1996). *Social Learning in Animals: The Roots of Culture*, chapter 1. Elsevier.
- Holden, C. and Mace, R. (1997). Phylogenetic analisys of the evolution of lactose digestion in adults. *Human Biology*.
- Ingold, T. (1986). Evolution and Social Life. Cambridge University Press.
- Jenkins, P. (1978). Cultural transmission of song patterns and dialect development in a free-living bird population. *Animal Behaviour*.
- Jong, K. D. (2002). Evolutionary Computation. Bradford.
- Kroeber, A. L. (1948). *Anthropology: Race, Language, culture, psychology, pre-history.* Harcourt, Brace and World.
- Laland, K. and Galef, B. G. (2009). *The Question of Animal Culture*. Harvard University Press.
- Laland, K. N. and Hoppitt, W. (2003). Do animals have culture? Evolutionary Anthropology.
- Laland, K. N. and Janik, V. M. (2006). The animal cultures debate. *Trends in Ecology and Evolution*.
- Lodish, H., Berk, A., Matsudaira, P., Kaiser, C., Krieger, M., Scott, M., Zipursky, S., and Darnell, J. (2004). *Molecular Biology of the Cell*. WH Freeman, 5 edition.
- McGrew, W. (1998). Culture in nonhuman primates. *Annual Review of Anthropology*.
- Rendell, L., Forgarty, L., and Lanland, K. (2010). Rogers' paradox recast and resolved: population structure and the evolution of social learning. *Evolution*.
- Rendell, L. and Whitehead, H. (2001). Culture in whales and dolphins. *Behavioral and Brain Sciences*, 24(02).
- Richerson, P. and Boyd., R. (2001). Culture is part of human biology: Why the superorganic concept serves the human sciences badly. In Maasen, S. and Winterhager, M., editors, *Science Studies: Probing the Dynamics of Scientific Knowledge*. Bielefeld: Verlag.
- Richerson, P. J. and Boyd, R. (2006). Not By Genes Alone: How Culture Transformed Human Evolution. University of Chicago Press.
- Rogers, A. (1988). Does biology constrain culture? *American Anthropologist*.

Gene-culture coevolution of language: measurement-interval dependence of evolutionary rate

Tsubasa Azumagakito^{1,2}, Reiji Suzuki¹ and Takaya Arita¹

¹Graduate School of Information Science, Nagoya University ²azumagakito@alife.cs.is.nagoya-u.ac.jp

Understanding the origin and evolution of language is key to understanding humans. The relationship between the biological evolution of language ability and the cultural evolution of language itself is extremely complex and shrouded in controversy because these mechanisms interact with each other despite the difference in their time scales (Mesoudi et al., 2011). The concept of coevolution between language and brain, in which language adapts to the brain and the brain adapts to language, is considered important in integrating biological and cultural evolution of humans (Deacon, 1997).

On the other hand, cultural linguistic change is often assumed to be much faster than biological change. For example, Chater et al. (2009) showed, using a computational model, that genetic natural selection may not keep pace with rapid language change. However, it is known that the rate of evolution has a dependence on the interval of time over which the rates are measured (Gingerich, 1993). Rates of evolution can be measured in darwins (d), which is a standardized unit of change in factors of e, the base of the natural logarithm, per millions of years (Gingerich, 1993).

$$d = (\ln(x_2) - \ln(x_1))/\Delta t \tag{1}$$

where x_1 is the mean trait value calculated at time t_1 , x_2 is the mean value at time t_2 , and Δt is the time interval between x_1 and x_2 . Fig. 1 shows an example of the measurement interval dependence of the evolutionary rate of a quantitative trait. If the evolutionary process is directional (a-i), the rates are stable irrespective of measurement time interval (a-ii). However, if the evolution process is less directional or fluctuating (b-i), the rates are inversely correlated with the measurement time interval (b-ii). This is because the fluctuation strongly affects the measured rate when the interval is short, while the general trend affects the rate when the interval is long. Perreault (2012) found that, by analyzing archaeological data, the rates of cultural evolution are inversely correlated with the measurement interval. He claimed that this explains why cultural change appears to be faster than biological change (because we tend to measure cultural evolution on a short time scale) but concluded that cultural evo-

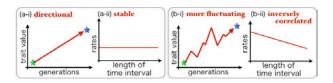


Figure 1: Examples of evolutionary trends for two quantitative traits and their associated evolutionary rates as a function of the measurement time interval.

lution is faster than biological evolution even when such a correlation is taken into account. This indicates the significance of this fine-grained comparison of evolutionary rates in the context of interactions between biological evolution of language ability and cultural evolution of language.

The purpose of this study is to clarify the relationship between the rates of biological evolution of language ability and cultural evolution of language, focusing on their measurement-interval dependence. In order to analyze this, we employ a bottom-up computational framework for investigating possible scenarios for gene-culture coevolution of language, which has been proposed in our previous works (Azumagakito et al., 2013, 2014).

In this model, there are finite number of agents and languages in a one-dimensional space, and agents can communicate with each other using their shared languages (Fig. 2). Each language is represented as a point in the space, and its distance from the origin (El) represents its expressiveness, which contributes to the expected fitness benefit of a successful communication using that language. Each agent is also represented as a point and an area surrounding the point. This point represents the agent's innate language ability determined by its gene Ea, and the area represents agent's linguistic plasticity determined by its gene P. The agent can use any languages within its plasticity area for communication, incurring a cost proportional to the size of the area. In each generation, all possible pairs of agents make an attempt to communicate. If two agents share one or more languages, they can communicate successfully. This definition means that agents who can communicate with many other agents



Figure 2: The linguistic space.

using expressive languages acquired with limited linguistic plasticity will have high fitness. After the communicative interactions of agents, biological evolution of language ability occurs according to a genetic algorithm.

Subsequently, the language population evolves according to four cultural processes: 1) Cultural change: the user of a language exerts a force that drags languages towards its location, 2) Division: a language is divided into two if the drag forces strongly pull it in opposing directions, 3) Extinction: any languages that are not used by any agents in one generation do not appear in the next generation, and 4) Fusion: when the distance between two languages is small enough, these languages are united into one language. Through the above processes, the agent population and the language population coevolve.

We conducted evolutionary experiments for 20,000 generations. The results showed that the agents and languages evolved to have high expressiveness through a cyclic coevolutionary process, which was discussed in our previous study (Azumagakito et al., 2014). Here, we measured evolutionary rates for languages (in terms of average El) and agents (in terms of average Ea).

Fig. 3 shows a typical coevolution process between (a) the average agent's trait and (b) language expressiveness, and their evolutionary rates measured over several time intervals. The x-axis ranges from 5000 to 20000 generations of evolution is not shown because there is no remarkable change of tendency. We see that both rates of evolution change dynamically and there are peaks for $\Delta t = 200$. These peaks correspond to rapid evolution of agents and languages. We also see that the rates of language evolution tended to be higher and fluctuate more than those of biological evolution in the case of $\Delta t = 200$. However, rates of language evolution became slower than biological rates when the time interval was long ($\Delta t \geq 1000$). In addition, Fig. 3 (c) shows linear approximations for both rates as a function of the time interval. The negative slopes clearly show their measurementinterval dependence. There was a statistically significant difference in the slopes (p < 0.05). Note that cultural rates were faster than biological rates when they were measured over a short time scale, but this reversed when the time interval was longer. This means that the measurement-interval dependence of the evolutionary rate of language is stronger than that of the rate of biological evolution. We propose that this is due to the lack of directionality in cultural evolution. This implies that biological evolution is more directional than cultural language evolution, and could therefore keep pace with language evolution.

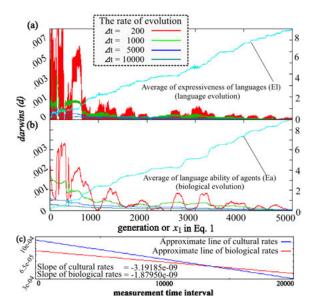


Figure 3: A typical coevolution process between (a) the average agent's trait and (b) language expressiveness, and their evolutionary rates measured over several time intervals. (c) Linear fits for both rates against the time interval over which the rates are measured.

Acknowledgements

This work was supported by Grant-in-Aid for JSPS Fellows (257917).

References

Azumagakito, T., Suzuki, R., and Arita., T. (2013). Cyclic behavior in gene-culture coevolution mediated by phenotypic plasticity in language. *Proceedings of the 12th European Conference on Artificial Life*, pages 617–624.

Azumagakito, T., Suzuki, R., and Arita., T. (2014). Does biological evolution keep pace with cultural evolution?: an analysis of gene-culture coevolution of language. *Proceedings of the 19th International Symposium on Artificial Life and Robotics*, pages 123–127.

Chater, N., Reali, F., and Christiansen, M. H. (2009). Restriction on biological adaptation in language evolution. *Proceedings* of the National Academy of Sciences, 106:1015–1020.

Deacon, T. W. (1997). The Symbolic Species: The Co-evolution of Language and Brain. W. W. Norton & Company, Inc.

Gingerich, P. D. (1993). Quantification and comparison of evolutionary rates. American Journal of Science, 293:453–478.

Mesoudi, A., Mcelligott, A. G., and Adger, D. (2011). Introduction: Integrating genetic and cultural evolutionary approaches to language. *Human Biology*, 83(2):141–151.

Perreault, C. (2012). The pace of cultural evolution. *PLoS ONE*, 7(9):e45150.

The Artificial Life-Form as Entrepreneur: Synthetic Organism-Enterprises and the Reconceptualization of Business

Matthew E. Gladden¹

¹Georgetown University, 37th & O St. NW, Washington, DC 20057 matthew.e.gladden@gmail.com

Extended Abstract

In this work we demonstrate the theoretical possibility and explore the implications of developing artificial life that functions as an autonomous business within the real-world human economy. By drawing on the Viable Systems Approach (VSA), we introduce the new concept of an "organismenterprise" that exists simultaneously as both a life-form and a business. We then reconceptualize the anthropocentric understanding of a "business" in a way that allows an artificial life-form to constitute a "synthetic" organism-enterprise (SOE) just as a human being functioning as a sole proprietor constitutes a "natural" organism-enterprise. Practical obstacles to the creation of SOEs are considered, along with possible means of surmounting them. SOEs would move a step beyond current examples of artificial life that produce goods or services within a simulated world or play a limited role within a human business: rather than competing against artificial organisms in a virtual world, SOEs could evolve through competition against human businesses in the real-world economy. We consider concrete examples of SOEs and conclude by highlighting legal, economic, and ethical issues that arise when a single economic ecosystem is shared by competing human and artificial life.

The concept of an "organism-enterprise." A business is defined as "the organized effort ... to produce and sell, for a profit, the goods and services that satisfy society's needs" (Pride, et al. 2014). Management theorists have drawn on biology to better understand the structure and function of such business organizations. Our research utilizes a systems theory grounded in neurophysiology, the Viable Systems Approach (VSA), that allows us to understand a business as an autopoietic organism or "system" that dwells within the ecosystem of a larger economy or "suprasystem" (Beer, 1981; Barile, et al. 2012). Within this ecosystem, a business must compete against other organisms for limited resources and adapt to environmental demands. In our human economy, individual businesses are born, grow, and die, and taken as a whole, this array of businesses forms an evolvable system.

We begin by considering one unique instance in which a business is not simply "analogous to" a living organism, but identical to it: namely, the case of a human being who functions as a sole proprietor. In this situation, a single system simultaneously satisfies all the requirements of being both a life-form and a business. Building on this case, we introduce the idea of a unitary "organism-enterprise," a concept that is already instantiated in the form of at least 20 million "human organism-enterprises" within the United States alone.

Reconceptualizing business to include synthetic organismenterprises. Utilizing VSA and the concept of an organismenterprise, we analyze the traditional anthropocentric understanding of business as an exclusively human activity to consider whether an artificial life-form could serve as a "synthetic organism-enterprise" (SOE) that is both a life-form and a business. We show that this is indeed possible, but requires us to transform our understanding of business.

For example, human businesses are traditionally described as requiring four kinds of resources: 1) human; 2) material; 3) financial; and 4) information. To replace this anthropocentric understanding, we propose that a business be understood more generally as requiring: 1) agent resources; 2) material resources; 3) value-storing media; and 4) information. Similarly, a human business requires functional units filling roles in production, finance, marketing, human capital management, and information technology. Drawing on VSA and the case of a human sole proprietor, we consider the ways in which these functions can be understood more generically, in such a way that they can also be performed by current and proposed forms of artificial life. We give particular attention to the role of "profit" in a human business and formulate an account of its correlate for an SOE: it is the difference between resources expended and received in exchanges in the suprasystem that provides an SOE with a potential for growth and insurance against environmental uncertainties.

Figure 1 provides an overview of our reconceptualized "business process cycle," which can be carried out equally well by either a human business or an artificial life-form that has been designed or evolved to fill a business role within a larger economic ecosystem.

Current obstacles to an artificial life-form as organismenterprise. Artificial life-forms have already been designed that are capable of carrying out this entire business process cycle within the simulated ecosystem of a virtual world (Kubera, et al. 2011). Similarly, there are artificial life-forms capable of carrying out parts of this cycle within human businesses in the economy of the "real world" (Kim and Cho, 2006). However, our survey of the field has not yet identified any existing artificial life-forms capable of carrying out this entire business process cycle within the real-world human economy. We identify a number of obstacles that currently prevent this from taking place, and we highlight those areas within the business process cycle that pose the greatest challenge for the future development of SOEs.

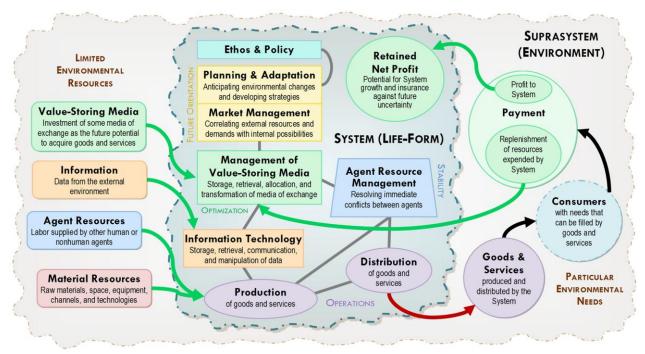


Figure 1. A reconceptualized business process cycle that applies to a human organism-enterprise as well as to a synthetic organism-enterprise (SOE) that has been designed or evolved to provide goods or services within some ecosystem.

Designing artificial life-forms that can complete the business process cycle. We briefly consider some approaches to overcoming these obstacles, so that an artificial life-form can not only meet the minimal requirements for constituting a synthetic organism-enterprise but potentially even excel in the role of entrepreneur (Ihrig, 2012). We especially consider the potential of virtual goods and cryptocurrencies to overcome the difficulty of providing an SOE with an effective means of utilizing value-storing media (Scarle, et al. 2012).

Evolution of artificial life through competition in the human economy. An SOE producing goods or services of value to human beings would be capable of competing against human businesses in the real-world economy. We consider whether these competitive pressures would be sufficient to drive evolution among SOEs. By utilizing the concept of "clockspeed" as a measure of the speed at which businesses must adapt and compete, we identify industries in which SOEs would likely evolve at an accelerated rate (Fine, 1998).

Specific examples and practical implications of artificial life as entrepreneur. An autonomous artificial life-form that is capable of securing all of the resources needed for survival and growth directly from the real-world human economy would in principle no longer be dependent on its human designer. Such possibilities are not risk-free: we imagine the case of computer viruses that are capable of evolving self-adaptive behavior rather than mere polymorphism or metamorphism (Beckmann, et al. 2009) and that no longer steal for the financial gain of human cybercriminals but to provide resources for their own survival, growth, and autonomously chosen pursuits. We also consider more optimistic cases, such as the development of artificial life-forms that build successful "careers" as artists or composers

or IT service-providers within the human economy and that are able to evolve in response to economic demands, without the active guidance or support of a human designer. Finally, we propose areas for future research to address the moral, legal, and economic issues that will arise from the existence of synthetic organism-enterprises and the fact that the productive and competitive capacities of successful SOEs could far surpass those of traditional human businesses.

References

Barile, S., Pels, J., Polese, F., Saviano, M. (2012). An Introduction to the Viable Systems Approach and Its Contribution to Marketing, *Journal of Business Market Management* 5(2):54–78.

Beckmann, B.E., Grabowski, L.M., McKinley, P.K., and Ofria, C. (2009).

Applying Digital Evolution to the Design of Self-Adaptive Software, *IEEE Symposium on Artificial Life*, 2009, pages 100–107.

Beer, S. (1981). *Brain of the Firm*. 2nd ed. John Wiley, New York.

Fine, C. (1998). Clockspeed: Using Business Genetics to Evolve Faster than Your Competitors. Perseus Books, Reading, MA.

Ihrig, M. (2012). Simulating Entrepreneurial Opportunity Recognition Processes: An Agent-Based and Knowledge-Driven Approach. In Byrski, A., Oplatková, Z., Carvalho, M., Kisiel-Dorohinicki, M., editors, Advances in Intelligent Modelling and Simulation: Simulation Tools and Applications, pages 27-54. Springer-Verlag, Berlin.

Kim, K., and Cho, S. (2006). A Comprehensive Overview of the Applications of Artificial Life, *Artificial Life* 12(1):153–82.

Kubera, Y., Mathieu, P., Picault, S. (2011). IODA: an interaction-oriented approach for multi-agent based simulation, Autonomous Agents and Multi-Agent Systems, 23(3):303-343.

Pride, W., Hughes, R., and Kapoor, J. (2014). Foundations of Business, 4e. Cengage Learning, Stamford, CT.

Scarle, S., Arnab, S., Dunwell, I., Petridis, P., Protopsaltis, A., De Freitas, S. (2012). E-commerce transactions in a virtual environment: virtual transactions, *Electronic Commerce Research*, 12(3):379-407.

Boolean Networks, Neural Networks and Machine Learning

On the Relationship between Local Rewiring Rules and Stationary Out-degree Distributions in Adaptive Random Boolean Network Models

Taichi Haruna¹ and Sayaka Tanaka¹

¹Department of Earth & Planetary Sciences, Graduate School of Science, Kobe University, 1-1, Rokkodaicho, Nada, Kobe 657-8501, Japan tharuna@penguin.kobe-u.ac.jp

Abstract

We discuss the relationship between local rewiring rules and stationary out-degree distributions in adaptive random Boolean network models that evolve toward criticality. We derive a theoretical formula for the relationship via a master equation approach. The theoretical result is shown to agree well with numerical simulation in three representative cases.

Introduction

Coupling between structural change of a network and dynamics on it is important to understand functioning of complex systems. Many models have been proposed to capture various types of couplings so far. They are called adaptive networks (Gross and Blasius, 2008; Gross and Sayama, 2009). One interesting type of the adaptive network model is the one in which network topology evolves toward dynamical criticality by a local rewiring rule. Several adaptive network models of this type have been studied numerically so far, for example, extremal dynamics on random networks (Christensen et al., 1998), random threshold networks (Bornholdt and Rohlf, 2000), neural networks (Bornholdt and Röhl, 2003; Meisel and Gross, 2009) and random Boolean networks (Liu and Bassler, 2006).

Random Boolean network (RBN) is a model of gene regulation networks (Kauffman, 1969, 1993). It has been shown that RBN exhibits a continuous phase transition from ordered to chaotic dynamics by a mean-field approximation (Derrida and Pomeau, 1986; Luque and Solé, 1997). In recent years, several approaches to map real-world networks to RBN have been proposed and shown that real-world networks are working close to criticality (Rämö et al., 2006; Balleza et al., 2008; Nykter et al., 2008). Biological significance of criticality has been also discussed: balance between robustness and evolvability (Aldana et al., 2007) and maximum information coordination (Ribeiro et al., 2008).

An RBN consists of a set of nodes connected via regulatory relationships represented as Boolean functions. Let the number of nodes be N and the in-degree of node i k_i . Each node can take two values 0 and 1 corresponding to off and on of a gene, respectively. For each node i, a random

Boolean function $f_i: \{0,1\}^{k_i} \to \{0,1\}$ is chosen. For each input $\mathbf{x} \in \{0,1\}^{k_i}$, the output of $f_i(\mathbf{x})$ is determined to be 1 with probability p and 0 with probability 1-p, where 0.5 is a parameter. Here, we consider the standardcase p = 0.5. The value of each node is updated by a given random Boolean function. For the time evolution of the whole system, we consider the classical synchronous updating scheme. Namely, we assume the existence of a global clock t and all nodes are updated synchronously. Let $x_i(t)$ be the value of node i at time step t. The state of the whole system is defined as $\mathbf{x}(t) = (x_1(t), \cdots, x_N(t))$. In the limit of large N, the dynamics of RBN is ordered if $z < \frac{1}{2p(1-p)}$, critical if $z=\frac{1}{2p(1-p)}$ and chaotic otherwise (Derrida and Pomeau, 1986; Luque and Solé, 1997), where z is the average in-degree of the underlying network of the RBN. When p = 0.5, the critical average in-degree is $z_c = 2$.

In this paper, we focus on adaptive random Boolean network models and discuss the relationship between local rewiring rules and the stationary out-degree distributions (Liu and Bassler, 2006). Real-world gene regulation networks have heavy-tailed out-degree distributions (Aldana et al., 2007). It is interesting problem whether we can reproduce this feature by an adaptive random Boolean network model that evolves toward criticality. (Liu and Bassler, 2006) reported that the stationary out-degree distribution is wider than the stationary Poisson-like in-degree distribution in their numerical simulation. However, its actual form is unknown. (MacArthur et al., 2010) studied a different type of adaptive network model that evolves toward criticality with a stationary heavy-tailed degree distribution. However, its mechanism is unclear. Here, we extend the model proposed by (Liu and Bassler, 2006) and obtain an theoretical expression for the relationship via a master equation approach. We also consider a new adaptive network model whose local rewiring rule is governed by local information transfer to examine whether our theoretical treatment is sensitive to the detail of local rewiring rules or not. We show that in both adaptive network models our theoretical result agrees well with numerical simulation for three representative cases: Poisson, exponential and truncated power law stationary out-degree distributions.

This paper is organized as follows. In section 2, we review two adaptive random Boolean network models considered in this paper. In section 3, we present the formula relating local rewiring rules and stationary out-degree distributions and compare it with numerical simulation. We derive it in Appendix. In section 4, we discuss meaning of the result and connection with real-world gene regulation network topology.

Adaptive network models

In this section, we introduce two adaptive random Boolean network models. The first one is an extension of the model proposed and numerically investigated by (Liu and Bassler, 2006). The second one is our own new model.

A model based on activity

The basic idea of the evolutionary rule in this model is *active* nodes tend to lose links, static nodes tend to get links (Bornholdt and Rohlf, 2000). After the dynamics of an RBN falls onto an attractor, a node is defined to be active if it changes its value on the attractor at least once. Otherwise, the node is static (Bornholdt and Rohlf, 2000; Liu and Bassler, 2006). The full algorithm can be described as follows:

- (i) The initial RBN with uniform in-degree $k_i = k_0$ for all node $i = 1, 2, \dots, N$ is generated.
- (ii) A random initial state $\mathbf{x}(0)$ is chosen. Iterate RBN dynamics until a dynamical attractor is reached. In the chaotic phase, it is hard to find an attractor in a realistic number of time steps. To cope with this problem, we follow the procedure adopted in (Liu and Bassler, 2006). We set the maximum attractor period $T_{\rm max}$ which we try to find. If no attractor is found in the first $2T_{\rm max}+T'$ time steps, we regard the last $T_{\rm max}$ steps as an attractor. Here, we take $T_{\rm max}=1000$ and T'=100 for efficient numerical simulation. It seems that the value of $T_{\rm max}$ does not affect the result of numerical simulation as long as it is sufficiently large. Indeed, we checked that $T_{\rm max}=500$ also reproduces the essentially the same results.

Let Γ be the period of the attractor.

(iii) A node i is randomly chosen. Calculate its average activity A(i) over the attractor:

$$A(i) = \frac{1}{\Gamma} \sum_{t=T}^{T+\Gamma-1} \mathbf{x}_i(t), \tag{1}$$

where we assume that the attractor is reached at least after T time steps. If 0 < A(i) < 1, then the node is called active. Otherwise, it is called static.

(iv) If chosen node i is active, then one of its incoming links is removed randomly. If it is static, then it gains a new incoming link. The source node of a new link is chosen by following a probability distribution depending on the outdegree of nodes.

(v) Boolean functions are randomly regenerated. Here, we use the method called annealed model in (Liu and Bassler, 2006): a new Boolean function is generated randomly for every node.

(vi) Go back to step (ii).

The time scale of network topology change (one cycle of step (ii) to step (v)) is called epoch. (Liu and Bassler, 2006) numerically shows that this evolutionary algorithm with the uniform probability distribution in step (iv) drives RBN toward criticality.

A model based on local information transfer

The local rewiring rule of this model depends on information transfer associated with links. To quantify information transfer, we use local transfer entropy (Lizier et al., 2008). Transfer entropy is a measure of the direction and magnitude of information transfer between two stationary stochastic processes (Schreiber, 2000). Let X, Y be stationary stochastic processes. The transfer entropy from Y to X is

$$T_{Y \to X} = \sum_{x(t)^{(k)}, x(t+1), y(t)^{(l)}} p(x(t+1), x(t)^{(k)}, y(t)^{(l)})$$

$$\times \log_2 \frac{p(x(t+1)|x(t)^{(k)}, y(t)^{(l)})}{p(x(t+1)|x(t)^{(k)})},$$
 (2)

where x(t) and y(t) are values of X and Y at time t, respectively, $x(t)^{(k)} = (x(t), x(t-1), \cdots, x(t-k+1))$ and $y(t)^{(l)} = (y(t), y(t-1), \cdots, y(t-l+1))$. Namely, the amount of information transfer from Y to X is quantified as the reduction of uncertainty to predict the future value of X from the present and past values of itself when one knows the present and past values of Y. In the following, we only consider the case k=l=1.

The local transfer entropy considers the log-ratio at the right hand side of (2). (Lizier et al., 2008) shows that the local transfer entropy works as a filter detecting coherent structures in complex spatiotemporal dynamics. Since the log-ratio can take a negative value, we can define misleading information transfer by adopting the local transfer entropy for information transfer quantification (Lizier et al., 2008).

Consider a node i in an RBN and neighboring nodes j_1, j_2, \dots, j_{k_i} that have a link to i. The local transfer entropy from node j_m to i at time t is defined by

$$lte(j_m \to i, t) = \log_2 \frac{p(x_i(t+1)|x_i(t), x_{j_m}(t))}{p(x_i(t+1)|x_i(t))}, \quad (3)$$

where the conditional probabilities at the right hand side are calculated from the frequency of each tuple of values from the all pairs $(i, j_n), n = 1, 2, \dots, k_i$ over the attractor to which the RBN dynamics reaches.

The idea for the evolutionary algorithm based on local information transfer is *links with misleading information transfer should be deleted*. Here, we implement this idea by

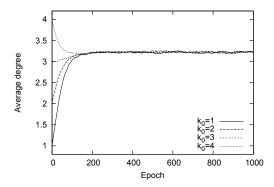


Figure 1: Time evolution of the average in-degree from different k_0 values in the adaptive network model based on local information transfer averaged over 100 trials. The system size is N=30.

replacing steps (iii) and (iv) in the model based on activity by the following (iii') and (iv').

(iii') Choose a node i and a time $T \leq t < T + \Gamma$ randomly. Calculate the local transfer entropy $lte(j_m \rightarrow i, t)$ for all $m = 1, 2, \cdots, k_i$.

(iv') If there is at least one link such that $lte(j_m \to i,t) < 0$, then the link with the smallest value of the local transfer entropy is deleted. If there are multiple such links, then one of them is chosen randomly and is deleted. If $lte(j_m \to i,t) \geq 0$ for all $m=1,2,\cdots,k_i$, then node i gets a new incoming link. The source node of a new link is chosen by following a probability distribution depending on the outdegree of nodes.

We numerically check that the model based on the local transfer entropy evolves toward criticality. Fig. 1 shows that time evolution of average in-degree from different k_0 values converges to a common value slightly above 3. Here, we adopt the uniform random choice of the source node of a newly added link at step (iv'). The deviation from the critical value 2 is rather large because of the finite size effect (N=30). Fig. 2 shows that the converged value of average in-degree approaches to the critical value 2 as N becomes larger. Indeed, it approaches algebraically to the criticality as we will see in next section (See Fig. 5).

Average in-degree evolves toward criticality in both adaptive RBN models. However, they have different stationary in-degree distributions as one can see in Fig. 3. On one hand, the stationary in-degree distribution for the model based on activity can be fitted by the Poisson distribution for all three link addition rules considered in next section. On the other hand, the width of that for the model based on local information transfer is strictly narrower than the Poisson distribution with the same average in-degree due to different local rewiring rules from those in the activity-based model.

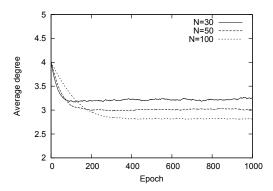


Figure 2: Time evolution of average in-degree for different system sizes N from the same k_0 value in the adaptive network model based on local information transfer. For each system size N, trajectories are averaged over 100 trials.

Main result

In this section, we present the formula relating local rewiring rules and stationary out-degree distributions. The theoretical result is tested against numerical simulation. The formula is derived in Appendix.

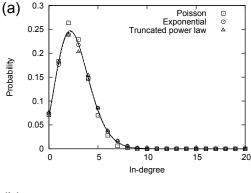
In the link addition part of the evolutionary algorithms described in section 2, the source node of a newly added link is chosen by following a probability distribution depending on the out-degree of nodes. Let $\sigma_{+,l}(e)$ be the probability that a particular node with out-degree l is chosen as the source of a new link given that the number of link increases at epoch e. In Appendix, we derive the following equation by analyzing the master equation describing time evolution of the out-degree distribution under the assumption that we can ignore the structure of networks except their degree distributions in the limit of large N:

$$\sigma_{+,l}^{s} = \frac{l+1}{Nz_{s}} \frac{P_{\text{out}}^{s}(l+1)}{P_{\text{out}}^{s}(l)},\tag{4}$$

where $\sigma^s_{+,l} = \lim_{e \to \infty} \sigma_{+,l}(e)$, z_s is the stationary average in-degree and $P^s_{\mathrm{out}}(l)$ is the stationary out-degree distribution. By (4), we can predict the link addition algorithm which produces the stationary out-degree distribution satisfying $P^s_{\mathrm{out}}(l) = P(l)$ for any given probability distribution P(l) with $\sum_{l=0}^{\infty} lP(l) = z_s$. Here, we study three representative cases.

Example 1 (Poisson distribution) If $P(l) = \frac{z_s^l}{l!} e^{-z_s}$, then $\frac{P(l+1)}{P(l)} = \frac{z_s}{l+1}$. It follows that $\sigma_{+,l}^s = \frac{1}{N}$ by (4). Thus, we expect the stationary Poisson out-degree distribution when we choose the source node of a newly added link uniformly at random.

Example 2 (Exponential distribution) If
$$P(l) = (1 - e^{-1/\kappa}) e^{-l/\kappa}$$
, where $\kappa = 1/\ln(1 + 1/z_s)$, then



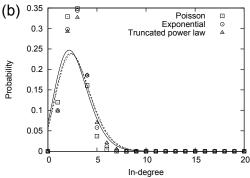
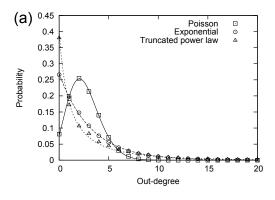


Figure 3: Stationary in-degree distributions for the two adaptive network models with different link addition rules introduced in section 2. (a) Results for the model based on activity. (b) Results for the model based on local information transfer. For every case, $N=200,\,k_0=4$ and the numerical stationary in-degree distribution is obtained by averaging in-degree distributions over 10000 epochs after 10000 initial transient epochs for a single trial. For reference, Poisson distribution with the same average degree is plotted for each case. Legends indicate different link addition algorithms described in section 3.

 $\frac{P(l+1)}{P(l)}=e^{-1/\kappa}.$ By (4), we have $\sigma_{+,l}^s=(l+1)\frac{e^{-1/\kappa}}{Nz_s}.$ The stationary exponential out-degree distribution is predicted when the source node of a newly added link is chosen in proportional to (l+1), where l is the out-degree of the node.

Example 3 (Truncated power law distribution) If

 $P(l)=C(l+1)^{-1}e^{-\lambda l}$, where C is a normalization constant and λ is determined by the equation $z_s=\sum_{l=0}^{\infty}lP(l)$, then $\frac{P(l+1)}{P(l)}=\frac{l+1}{l+2}$. By (4), $\sigma_{+,l}^s=\frac{(l+1)^2}{l+2}\frac{1}{Nz_s}$. If the source node of a newly added link is chosen in proportional to $\frac{(l+1)^2}{l+2}$, where l is the out-degree of the node, then the theory predicts that we have the stationary truncated power law out-degree distribution.



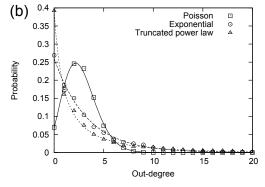
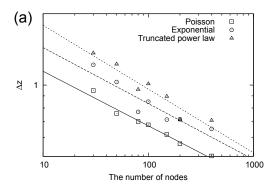


Figure 4: Stationary out-degree distributions for the two adaptive network models with different link addition rules. Theoretical lines are compared with the numerical distributions. (a) Results for the model based on activity. (b) Results for the model based on local information transfer. The parameters and the averaging procedure are the same as in Fig. 3.

We compare the numerical stationary out-degree distributions with the theoretical ones for both models of adaptive RBN in Fig. 4. In every case, the theoretical line agrees well with the numerical result. Fig. 5 shows that the stationary average in-degree approaches algebraically to the critical value 2 as the system size N increases irrespective of models and link addition rules though the speed of convergence is case-by-case. These results suggests that we can control the stationary out-degree distribution by (4) while keeping evolution toward criticality.

Discussion

In this paper, we have shown that we can control the stationary out-degree distribution of two adaptive RBN models that evolve toward criticality by modifying the link addition part of of the local rewiring rule. Only the information of neighboring nodes that have a link to a node to be removed is used for the link deletion part of the rule. In this sense, the link deletion part is informationally local. On the oth-



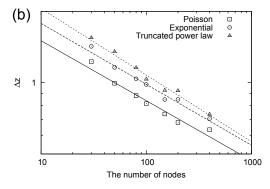


Figure 5: System size dependence of deviation from criticality $\Delta z=z_s-2$. (a) Results for the model based on activity. The slopes of the lines are -0.230 (real, Poisson), -0.236 (dashed, Exponential) and -0.269 (dotted, Truncated power law). (b) Results for the model based on local information transfer. The slopes of the lines are -0.253 (real, Poisson), -0.257 (dashed, Exponential) and -0.281 (dotted, Truncated power law). For every case, the stationary average indegree z_s is calculated by averaging average in-degrees over 10000 epochs after 10000 transient initial epochs in a single run from $k_0=4$.

er hand, the link addition part is informationally global: it inevitably needs global information because the source node of a newly added link is chosen from the set of all nodes in the whole system although the resulting structural change of network topology is local. Thus, there is asymmetry between the link deletion part and the link addition part in terms of information used. Our result indicates that evolution toward criticality can be achieved irrespective of the form of informationally global link addition part.

It has been suggested that real-world gene regulation networks have heavy-tailed out-degree distributions (Aldana et al., 2007). Assuming that the local rewiring rule studied in this paper is a good approximation to realistic network evolutionary process, the emergence of heavy-tailed out-degree distributions in real-world gene regulation networks cannot be directly associated with evolution toward

dynamical criticality. Rather, it could be a consequence of an unknown macroscopic constraint. Some models that evolve toward criticality and also generate a heavy-tailed degree distribution have been proposed so far (MacArthur et al., 2010; Torres-Sosa et al., 2012). In these models, mechanism to generate a heavy-tailed degree distribution would be involved implicitly in a whole evolutionary rule that takes into account global properties of the system. In contrast, the mechanism related to evolution toward criticality and that generates a given out-degree distribution can be separated explicitly in the adaptive network models studied in this paper.

If a macroscopic constraint forces a specific form of stationary out-degree distribution, then the underlying local rewiring rule is uniquely selected via (4). What macroscopic constraints can give rise to heavy-tailed out-degree distributions found in real-world gene regulation networks? Answering this question is out of the scope of the present paper and we leave it for future work.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 25280091.

Appendix

In this Appendix, we derive equation (4).

Let p_k be the probability that a chosen node with indegree k gets a new incoming link. The time evolution of in-degree distribution $P_{\rm in}(k,e)$ is described by the master equation $P_{\rm in}(k,e+1)=\left(1-\frac{1}{N}\right)P_{\rm in}(k,e)+\frac{1}{N}\left(p_{k-1}P_{\rm in}(k-1,e)+(1-p_{k+1})P_{\rm in}(k+1,e)\right)$ for $k\geq 1$ and $P_{\rm in}(0,e+1)=\left(1-\frac{1}{N}\right)P_{\rm in}(0,e)+\frac{1}{N}(1-p_1)P_{\rm in}(1,e).$

By equating $P_{\rm in}(k,e+1)$ to $P_{\rm in}(k,e)$, we obtain the condition for the stationary in-degree distribution $P_{\rm in}^s(k)$:

$$(1 - p_{k+1})P_{\text{in}}^{s}(k+1) = p_k P_{\text{in}}^{s}(k)$$
 (5)

for all $k \ge 0$. Here, we only assume that the existence of p_k and we leave the problem how to calculate p_k for a future work. It could be obtained under a certain mean-field approximation of the models. In the following, we concentrate on the analysis of the stationary out-degree distribution.

The time evolution of out-degree distribution $P_{\mathrm{out}}(l,e)$ follows the following master equation: $P_{\mathrm{out}}(l,e+1) = (1-q_{+,l}(e)-q_{-,l}(e)) P_{\mathrm{out}}(l,e) + q_{+,l-1}(e) P_{\mathrm{out}}(l-1,e) + q_{-,l+1}(e) P_{\mathrm{out}}(l+1,e)$ for $l \geq 1$ and $P_{\mathrm{out}}(0,e+1) = (1-q_{+,0}(e)) P_{\mathrm{out}}(0,e) + q_{-,1}(e) P_{\mathrm{out}}(1,e)$, where $q_{+,l}(e)$ is the probability that a node with out-degree l gets a new outgoing link at epoch e and $q_{-,l}(e)$ is the probability that a node with out-degree l loses an outgoing link at epoch e.

Put $\rho_+(e) := \sum_{k=0}^{\infty} P_{\mathrm{in}}(k,e) p_k$ and $\rho_-(e) := \sum_{k=0}^{\infty} P_{\mathrm{in}}(k,e) (1-p_k)$. Let $\sigma_{+,l}(e)$ be the probability that a particular node with out-degree l is chosen as the source of

a new link given that the number of link increases at epoch e and $\sigma_{-,l}(e)$ the probability that an outgoing link of a particular node with out-degree l is deleted given that the number of link decreases at epoch e. By definitions of ρ and σ , we have

$$q_{+,l}(e) = \rho_{+}(e)\sigma_{+,l}(e)$$
 (6)

and

$$q_{-,l}(e) = \rho_{-}(e)\sigma_{-,l}(e).$$
 (7)

We now introduce the configuration model (random networks with a specified degree distribution (Newman et al., 2001)) approximation: links are randomly shuffled while keeping the degree of each node at the end of each epoch. Then, we have

$$\sigma_{-,l}(e) = \frac{l}{Nz(e)} \tag{8}$$

in the limit of large N, where z(e) is the average in-degree at epoch e (which is equal to the average out-degree). Indeed, suppose that the chosen node at epoch e has in-degree k and it loses one of its incoming link. Noting that Nz(e) is the number of arcs, we find that the probability that there is a link from a node with out-degree l to a node with in-degree l is $1 - \left(1 - \frac{l}{Nz(e)}\right)^k \approx 1 - \left(1 - \frac{kl}{Nz(e)}\right) = \frac{kl}{Nz(e)}$ in the limit of large l. The probability of deletion of this link is l/k under the approximation. Thus, we obtain $\sigma_{-,l}(e) = \frac{kl}{Nz(e)}\frac{1}{k} = \frac{l}{Nz(e)}$.

As in the case with the stationary in-degree distribution, the stationary out-degree distribution $P_{\text{out}}^s(l)$ should satisfy

$$q_{-l+1}^s P_{\text{out}}^s(l+1) = q_{+l}^s P_{\text{out}}^s(l),$$
 (9)

for all $l\geq 0$, where $q_{+,l}^s=\rho_+^s\sigma_{+,l}^s$, $q_{-,l}^s=\rho_-^s\sigma_{-,l}^s$, $\rho_+^s=\sum_{k=0}^\infty P_{\rm in}^s(k)p_k,$ $\rho_-^s=\sum_{k=0}^\infty P_{\rm in}^s(k)(1-p_k)$ and $\sigma_{+,l}^s$ and $\sigma_{-,l}^s$ are stationary values for corresponding epoch-dependent quantities, respectively. Since $p_0=1$, we have

$$\rho_{+}^{s} = \rho_{-}^{s} \tag{10}$$

by (5). Substituting the stationary analogue of (8) and (10) into (9), we obtain equation (4)

$$\sigma_{+,l}^s = \frac{l+1}{Nz_s} \frac{P_{\text{out}}^s(l+1)}{P_{\text{out}}^s(l)},$$

where z_s is the stationary average in-degree.

References

- Aldana, M., Balleza, E., Kauffman, S., and Resendiz, O. (2007). Robustness and evolvability in genetic regulatory networks. *Journal of Theoretical Biology*, 245:433–448.
- Balleza, E., Alvarez-Buylla, E. R., Chaos, A., Kauffman, S., Shmulevich, I., and Aldana, M. (2008). Critical dynamics in genetic regulatory networks: Examples from four kingdoms. *PLOS ONE*, 3:e2456.

- Bornholdt, S. and Röhl, T. (2003). Self-organized critical neural networks. *Physical Review E*, 67:066118.
- Bornholdt, S. and Rohlf, T. (2000). Topological evolution of dynamical networks: Global criticality from local dynamics. Physical Review Letters, 84:6114–6117.
- Christensen, K., Donangelo, R., Koiller, B., and Sneppen, K. (1998). Evolution of random networks. *Physical Review Letters*, 81:2380–2383.
- Derrida, B. and Pomeau, Y. (1986). Random networks of automata: A simple annealed approximation. *Europhysics Letters*, 1:45–49.
- Gross, T. and Blasius, B. (2008). Adaptive coevolutionary networks: a review. *Journal of The Royal Society Interface*, 5:259–271.
- Gross, T. and Sayama, H., editors (2009). Adaptive Networks: Theory, Models and Applications. Springer Verlag, Heidelberg.
- Kauffman, S. A. (1969). Metablic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biol*ogy, 22:437–467.
- Kauffman, S. A. (1993). The Origins of Order: Self-Organization and Selection in Evolution. Oxford University Press, New York.
- Liu, M. and Bassler, K. E. (2006). Emergent criticality from coevolution in random boolean networks. *Physical Review E*, 74:041910.
- Lizier, J. T., Prokopenko, M., and Zomaya, A. Y. (2008). Local information transfer as a spatiotemporal filter for complex systems. *Physical Review E*, 77:026110.
- Luque, B. and Solé, R. V. (1997). Phase transitions in random networks: Simple analytic determination of critical points. *Physical Review E*, 55:257–260.
- MacArthur, B. D., J., S.-G. R., and Ma'ayan, A. (2010). Microdynamics and criticality of adaptive regulatory networks. *Physical Review Letters*, 104:168701.
- Meisel, C. and Gross, T. (2009). Adaptive self-organization in a realistic neural network model. *Physical Review E*, 80:061917.
- Newman, M. E. J., Strogatz, S. H., and Watts, D. J. (2001). Random graphs with arbitrary degree distributions and their applications. *Phys. Rev. E*, 64:026118.
- Nykter, M., Price, N., Aldana, M., Ramsey, S. A., Kauffman, S. A., Hood, L. E., Yli-Harja, O., and Shmulevich, I. (2008). Gene expression dynamics in the macrophage exhibit criticality. *Proceedings of National Academy of Science, U.S.A.*, 105:1897–1900.
- Rämö, P., Kesseli, J., and Yli-Harja, O. (2006). Perturbation avalanches and criticality in gene regulatory networks. *Journal of Theoretical Biology*, 242:164–170.
- Ribeiro, A. S., Kauffman, S. A., Lloyd-Price, J., Samuelsson, B., and Socolar, J. E. S. (2008). Mutual information in random boolean models of regulatory networks. *Physical Review E*, 77:011901.

- Schreiber, T. (2000). Measuring information transfer. *Physical Review Letters*, 85:461–464.
- Torres-Sosa, C., Huang, S., and Aldana, M. (2012). Criticality is an emergent property of genetic networks that exhibit evolvability. *PLOS Computational Biology*, 8:e1002669.

Random Fuzzy Networks

Octavio B. Zapata¹ and Carlos Gershenson¹

¹Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, UNAM, México, D.F. 01000 ocbzapata@gmail.com cgg@unam.mx

Introduction

No model can be considered effective if fundamentally it is more complicated than what it's trying to represent. However, extreme simplification may potentially overlook important non-primary features, or even neglect the possibility to represent ambiguous or unclear observations. Therefore, achieving a balance between parsimonious and detailed models is of utmost importance for science and engineering.

Over the past few decades, random Boolean networks (RBNs) (Kauffman, 1969) have become popular models for genetic regulatory networks. This popularity is associated with the fact that RBNs are very general models. No functionality or structure is particularly assumed when constructing them. However, the Boolean idealization has been constantly criticized based on the assumption that constraining the variables of the model to have only two possible values (0 and 1) entails a loss of dynamical information in the analysis of real gene expression data.

Boolean and Continuous Models

To randomly generate a network of dynamically interacting agents is very convenient for modeling systems where the underlying structure or performance is not yet explicitly recognized. The RBN model consists of n nodes, each of which have k randomly chosen input links, that determine its value $\sigma_i(t)$ at time step t from the set $\Sigma = \{0,1\}$ where $i=1,\ldots,n$. At time t+1 the state $\sigma_i(t+1)$ is completely determined by a randomly chosen Boolean function $f_i:\Sigma^k\to\Sigma$ from its k inputs at a previous time step, i.e. $\sigma_i(t+1)=f_i(\sigma_{i_1}(t),\ldots,\sigma_{i_k}(t))$. These functions can be computed via a lookup table —one for each function.

There have been several critiques for both, the Boolean idealization (Yingjun et al., 2007; Wittmann et al., 2009), and the temporal discretization (Bagley and Glass, 1996; Kappler et al., 2003) of RBNs. On the other hand, methods that justify the Boolean case have been proven effective, e.g., in Shmulevich and Zhang (2002); Karlebach (2013).

Multi-state approaches (Wuensche, 1998; Solé et al., 2000; Wittmann and Theis, 2011) emerged from the consideration that in some natural systems, the discreteness of

the actions performed at individual levels is well defined and thus, the dynamic assumption from a discrete model is fairly plausible; nevertheless, transitions from one state to another may not be sufficiently sharp for being represented by Boolean variables. Multi-valued variables that range over multiple (not only binary) states enhance the capability of the model to accurately portray gradual changes on the dynamic individual behavior of the system.

Random Fuzzy Networks

We extended the classical Boolean model (Kauffman, 1969; Wuensche, 1998; Aldana et al., 2003; Gershenson, 2004) by using concepts from fuzzy logic (Zadeh, 1965). The number of possible states s in which a node may be, vary according to the parameter $\phi \in [0,1]$. A random fuzzy network (RFN) is a RBN where the initial state of each node can take values from the set $\Sigma = \{1, s-2/s-1, s-3/s-1, \ldots, 0\}$ whereas s depends on ϕ . The fuzzyness of the state space—and therefore of the net—can be controlled by ranging ϕ along the interval [0,1]. More specifically, when $\phi=0$, we have the Boolean case where s=2. Otherwise, if $\phi=1$, then $s\to\infty$ and states can take values from all rational numbers between zero and one. We call s the base of the net. Whichever base may be achieved choosing the right ϕ .

Each lookup table in a RBN can be fuzzyfied, representing it in terms of combinations of AND, OR, and NOT and using Zadeh's operators (MIN, MAX, and 1-x, respectively). In this way, depending only on the diversity of the initial states, the same RFN can go from Boolean to continuous.

We introduce the concept of a *family of attractors*, to group attractors with the same transition schema. This concept helps us to study the state space dynamics of RFNs, regardless of its base. Specific state values are not taken into account. A family represents in which *direction* the states within an attractor change, i.e., a node state either goes up (represented by 1) if $\sigma_i(t) < \sigma_i(t+1)$, down(-1) if $\sigma_i(t) > \sigma_i(t+1)$ or stays(0) if $\sigma_i(t) = \sigma_i(t+1)$. All attractors on a particular family have the same length. Different attractors might belong to the same family. In particular,

steady states (point attractors) are all members of the family, e.g., (0, ..., 0). Transitions space size (3^n) is fixed for all ϕ .

Preliminary Results

The extended RFN model was implemented on Java virtual laboratory RBNLab (Gershenson, 2005). We explore the state space dynamics on ensembles $\mathbf{F}(n,k,\phi)$ of fuzzy nets with n=10,20,100 and $k=1,\ldots,5$. Five different ϕ values were chosen, giving base numbers s equals to 2,4,8,16 and ∞ , respectively. The number of initial states explored was arbitrarily bounded because as we let $\phi \to 1$, the size of the state space (s^n) increases proportionally to the base—and it rapidly becomes intractable. We do not consider significant the results for n=10 ensembles. On the other hand, due to the potentially astronomical amount of system states for n=100 nets, results might be biased because with a high probability one does not explore a representative part of the state space.

As shown in figure 1, the average number of attractors $\langle att \rangle$ found for n=20 RFNs, increases exponentially for s>2. The average number of families $\langle fam \rangle$ also increases but the growth-rate seems to be linear. For $\langle att \rangle$, standard deviations were maximal on continuous ($\phi=1$) ensembles, and for $\langle fam \rangle$ they were when k=5. We did not observe a significant increase on $\langle fam \rangle$ when data representation goes from Boolean to continuous. This result suggests that in the process of discretization there is few loss of information (families), and hence we can assert that fuzzy networks are capable to achieve a significant degree of complexity reduction –via its ϕ parameter. Therefore, we conclude by saying that the use of Boolean models for coarsely studying the discrete dynamics of real-valued observations is further supported.

Acknowledgment

Ankita Panwar contributed to an initial stage of this research. We thank three anonymous referees for their valuable comments.

References

- Aldana, M., Coppersmith, S., and Kadanoff, L. P. (2003). Boolean dynamics with random couplings. In Kaplan, E. e. a., editor, Perspectives and Problems in Nonlinear Science, pages 23– 89
- Bagley, R. J. and Glass, L. (1996). Counting and classifying attractors in high dimensional dynamical systems. *Journal of Theoretical Biology*, 183:269–284.
- Gershenson, C. (2004). Introduction to random boolean networks. In Bedau, M. e. a., editor, *ALife IX Workshop and Tutorial Proceedings*, pages 160–173.
- Gershenson, C. (2005). RBNLab. http://rbn.sourceforge.net.
- Kappler, K., Edwards, R., and Glass, L. (2003). Dynamics in high-dimensional model gene networks. *Signal Processing*, 83:789–798.

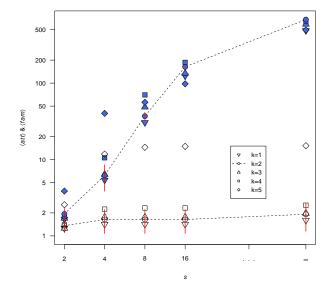


Figure 1: Average numbers of attractors $\langle att \rangle$ and average number of families $\langle fam \rangle$ found in ensembles with n=20, $k=1,\ldots,5$ and $0 \leq \phi \leq 1$. Shaded symbols represent $\langle att \rangle$ and empty symbols represent $\langle fam \rangle$. Standard deviations shown only for k=2. Note the logarithmic scales.

- Karlebach, G. (2013). Inferring boolean network states from partial information. *EURASIP Journal on Bioinformatics and Systems Biology*, 2013 (11).
- Kauffman, S. A. (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biol*ogy, 22:437–467.
- Shmulevich, I. and Zhang, W. (2002). Binary analysis and optimization-based normalization of gene expression data. *Bioinformatics*, 18 (4):555–565.
- Solé, R. V., Luque, B., and Kauffman, S. (2000). Phase transition in random networks with multiple states. Technical Report 00-02-011, Santa Fe Institute.
- Wittmann, D. M., Krumsiek, J., Saez-Rodriguez, J., Lauffenburger, D. A., Klamt, S., and Theis, F. J. (2009). Transforming boolean models to continuous models. *BMC Syst Biol*, 3.
- Wittmann, D. M. and Theis, F. J. (2011). Dynamic regimes of random fuzzy logic networks. *New Journal of Physics*, 13.
- Wuensche, A. (1998). Discrete dynamical networks and their attractor basins. In Standish, R. e. a., editor, *Complex Systems* '98, pages 3–21.
- Yingjun, C., Wang, P. P., and Resconi, G. (2007). Reverse engineering of the nk boolean network and its extension fuzzy logic network. *New Mathematics and Natural Computation*, 3 (1):69–87.
- Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8:338–353.

Structure and dynamics affect the controllability of complex systems: a Preliminary Study

Alexander J. Gates^{1,2} and Luis M. Rocha^{1,2,3}
¹School of Informatics and Computing, Indiana University, Bloomington, IN 47406, USA
²Cognitive Science Program, Indiana University, Bloomington, IN 47406, USA
³Instituto Gulbenkian de Ciencia, Oeiras, Portugal

Complex systems are typically understood as large nonlinear multivariate systems. Their organization and behavior are commonly modeled by representations such as graphs and automata networks. Graphs, where nodes representing variables lack intrinsic dynamics, capture the *structure* or organization of complex systems. The simplest way to study multi-variate *dynamics*, is to allow network nodes to have states and update them with automata; for instance, Boolean networks (BN) are canonical models of complex systems and exhibit a wide range of dynamical behaviors [1].

The structure of networks has provided many insights into the organization of complex systems [2]. The success of this approach is its ability to capture the organization of complex systems, and how it changes in time (network evolution) without explicit dynamical rules for node variables. As the field matures, however, there is a need to move from understanding to controlling complex systems. This is particularly true in systems biology and medicine, where increasingly accurate models of biochemical regulation have been produced [3]. More than understanding the organization of biochemical regulation, we need to derive control strategies that allow us, for instance, to move a mutant cell to a wild-type state [4], or revert a mature cell to a pluripotent state [5]. Towards these goals, a question of central importance remains: How well does network structure represent the multivariate dynamics of the underlying complex system, especially from the point of view of control?

Network controllability aims precisely to identify a minimal set of *driver variables* (a.k.a. driver nodes) from the structural network, which can fully control system dynamics—i.e. drive system dynamics to any state-space configuration [6]. *Structural controllability* is an influential method to derive driver variables, using only structural properties of the system without any consideration of its dynamical details [7]. It has been used to suggest, for instance, that biological systems are harder to control than social systems [8]. However, applications of structural controllability have been heavily critiqued due to its stringent assumptions [9].

Here we explore the relationship between network structure and controllability through the analysis of dynamical ensembles of BN. The control problem for general BN is computationally intractable (NP-hard) [10]. Simplification techniques, such as structural controllability or those based on dynamical redundancy [4], are thus highly desirable. The BN studied here are discrete dynamical systems $\mathcal{X} = \{x_i\}$ of N Boolean variables $x_i \in \{0,1\}$ that are updated synchronously according to deterministic logical functions. The structural network specifies all directed pairwise interactions $\{e_{ij}\}$ which indicate when variable x_i is an input for the logic of x_i . At time t, the network is in a configuration X^t , which is the vector of all variable states $(x_i(t))$ at time t. The overall dynamics of temporal sequences of network configurations can be represented by the state-transition graph (STG). In this graph, each node is a configuration X^t of the BN and each directed edge is a transition from X^t to X^{t+1} . Thus, the STG fully describes the 2^N possible configurations and transitions in the network's dynamical landscape.

We study the control due to a subset $\mathcal{D}\subset\mathcal{X}$ of driver variables as instantaneous perturbations to the variable's state. To capture all possible dynamically allowable trajectories due to controlled interventions on \mathcal{D} , we introduce the *controlled state transition graph* (CSTG $_{\mathcal{D}}$). The CSTG $_{\mathcal{D}}$ is an extension of the STG where additional edges connect a configuration to each of its $2^{|\mathcal{D}|}-1$ perturbed counterparts. The network is fully controllable if a trajectory exists between every pair of configurations; for BN this is equivalent to requiring the CSTG $_{\mathcal{D}}$ to be strongly-connected.

We extend this binary notion of controllability by tallying the fraction of configurations that are controlled by driver set \mathcal{D} . Given a specific configuration X_k , the fraction of reachable configurations $r_{\text{CSTG}_{\mathcal{D}}}(X_k)$ is the number of other configurations reachable on graph $\text{CSTG}_{\mathcal{D}}$ via a directed path starting from X_k , normalized by 2^N-1 . The mean fraction of reachable configurations is then given by $\overline{R} = \langle r_{\text{CSTG}_{\mathcal{D}}}(X_k) \rangle_k$, where $k=1\dots 2^N$. It measures the fraction of configurations which are on average reachable by controlling the variables in \mathcal{D} . The mean fraction of controlled configurations is $\overline{C} = \langle r_{\text{CSTG}_{\mathcal{D}}}(X_k) - r_{\text{STG}}(X_k) \rangle_k$. It measures the average fraction of controlled configurations by discounting those transitions which would have naturally

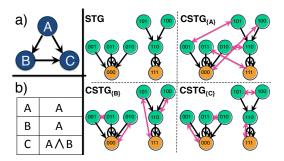


Figure 1: a) FeedForward network motif structure with dynamics given by the rules in b) give rise to the State-Transition Graph (STG). Each of the singleton sets $\{A\}$, $\{B\}$, and $\{C\}$ are used as driver variable sets to produce the Controlled STGs (CSTG $_{\mathcal{D}}$) shown.

occurred due to the uncontrolled dynamics of the network. A fully controlled network must have $\overline{R}=1.0$, but partially controlled networks will vary in [0,1].

We start by characterizing the entire ensemble of possible BN dynamics for the N=3 variable Feed-Forward network motif [11] shown in Figure 1a. For each network structure, there are $L = \sum_{i} 2^{k_i}$ possible transition rules, where k_i is the in-degree of variable x_i . In this case, the full ensemble consists of $2^L = 2^6 = 64$ distinct networks; the logic of one is depicted in Figure 1b. This figure also depicts its STG and the CSTG $_{\mathcal{D}}$ for various driver sets \mathcal{D} . If we additionally remove all variable transition functions that refer to tautologies, contradictions, and functions always controlled by a single input (fully canalizing [12]), we obtain a smaller Non-trivial ensemble. The controllability analysis for both ensembles is shown in Figure 2. Notice that structural controllability analysis predicts that variable A is capable of fully controlling the network. However, this driver variable $(CSTG_{\{A\}})$ fails to control a large majority of the possible BN. Even pairs of variables cannot fully control all networks $(\overline{R} < 1.0)$; to guarantee full control for every network, all three variables need to be driven.

This simple example highlights the tenuous relationship between structure and dynamics for complex systems, and the implications for understanding and characterizing their control. This work has been extended in four significant ways [13], and will be showcased at the conference: (1) analysis of several more network motifs; (2) extend our measures of partial control to the more biologically relevant concept of attractor control (since all non-attractor configurations are transients); (3) study the space of random BN (such as Erdos-Renyi structural networks with transitions parameterized by in-degree and bias) to establish that similar results hold as the systems are scaled up; (4) study models of biochemical regulation such as the segment polarity generegulatory BN of Drosophila melanogaster [14], the control of which is known [4]. These studies show how the structure-only analysis of complex systems tends to fail to

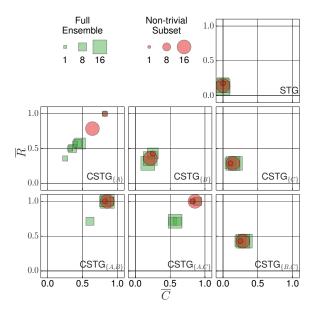


Figure 2: Measures \overline{R} and \overline{C} for the Full Ensemble of 64 BN (green squares and red circles) with structure given by the Feed-Forward network motif shown in Figure 1, as controlled by various combinations of driver variables. The Non-trivial subset is highlighted by red circles.

properly characterize their full or partial controllability. Additionally, we lay the groundwork for understanding which restrictions must be enforced on the transition functions of BN, such that structure may suffice for predicting controllability. Our full study and analysis has been submitted for publication [13].

References

- [1] Kauffman, S. A. (1993) *The Origins of Order: self-organization and selection in evolution.* (Oxford University Press).
- [2] Boccaletti, S, Latora, V, Moreno, Y, Chavez, M, & Hwang, D.-U. (2006). *Physics Reports* **424**, 175–308.
- [3] R, Albert & S.M. Assmann. (2009). *Plant Systems Biology* **35**, 41–66.
- [4] Marques-Pita, M & Rocha, L. M. (2013). PLOS ONE 8, e55946.
- [5] Wang, J, Zhang, K, Xu, L, & Wang, E. (2011). PNAS 108, 8257–8262.
- [6] Valente, T. (2012). Science 337, 49-53.
- [7] Liu, Y.-Y, Slotine, J.-J, & Barabási, A.-L. (2011). Nature 473, 167–173.
- [8] Egerstedt, M. (2011). Nature 473, 158-159.
- [9] Cowan, N. J, Chastain, E. J, Vilhena, D. A, Freudenberg, J. S, & Bergstrom, C. T. (2012). *PLOS ONE* 7, e38398.
- [10] Akutsu, T, Hayashida, M, Ching, W.-K, & Ng, M. (2007). J. Theor. Biol. 244, 670–679.
- [11] Alon, U. (2007). Nature Reviews Genetics 8, 450-461.
- [12] Kauffman, S, Peterson, C, Samuelsson, B, & Troein, C. (2004). PNAS 101, 17102–17107.
- [13] Gates, A. J & Rocha, L. M. (2014) Control of complex networks requires structure and dynamics. *submitted*.
- [14] Albert, R & Othmer, H. (2003). J. Theor. Biol. 223, 1–18.

Evolving Spiking Networks for Turbulence-Tolerant Quadrotor Control

David Howard¹ and Alberto Elfes¹

¹CSIRO, Autonomous Systems Program, 1 Technology Court Pullenvale, QLD 4069 Australia david.howard@csiro.au

Abstract

We investigate the automatic development of robust quadrotor neurocontrollers based on spiking neural networks. A self-adaptive evolutionary algorithm is used to generate high-utility topology/weight combinations in the networks, and a simple synaptic plasticity mechanism provides some degree of in-trial adaptation. Incremental evolution gradually increases the severity of environmental conditions that the agent can successfully handle. Results compare the spiking networks to tuned Proportional/Integral/Derivative controllers and feedforward neural networks for waypoint-holding experiments in varied atmospheric conditions. It is shown that the spiking controllers are able to maintain a closer distance to the waypoint than the comparative controllers, and more effectively deal with more challenging environmental conditions.

Introduction

The control of quadrotors in outdoor scenarios is a difficult task as both the agent and environment are highly dynamic. Traditional controllers, e.g. Proportional/Integral/Derivative (PID) (Minorsky, 1922), are still widely used in such scenarios, yet they are restricted to linear control and often require hand-tuning to achieve acceptable performance. Recent research has shown that nonstandard nonlinear controllers such as evolved feedforward neural networks (Rumelhart and McClelland, 1986) can automatically generate controllers that provide performance enhancements (Shepherd and Tumer, 2010).

Due to the dynamical nature of the task, *stateful* neural network representations may make suitable controllers due to their ability to (i) generate highly nonlinear action sequences through internal network dynamics, and (ii) harness latent information that may be found in temporally sequenced states (Maass and Zador, 1999). In this paper, we focus on the evolution of stateful spiking neural controlelrs (Gerstner and Kistler, 2002) for outdoor scenarios.

Spiking control systems offer a plethora of benefits, including temporally-sensitive neuron memory, the ability to perform highly nonlinear state-action mappings which may be demanded by the platform, and relatively straightforward

mapping to power-efficient pulsed hardware implementations (important when considering future onboard control). Spiking networks are biologically plausible representations and as such permit numerous extensions based on neuroscientific principles — a simple model of synapse-level Hebbian plasticity (Hebb, 1949) here provides in-trial adaptivity through spike timing coincidences. Action calculation is also based on spike coincidences (at the output neurons).

A Genetic Algorthm (GA) (Holland, 1975) is used to automatically generate high-performance controllers by seeking beneficial network compositions — the weights and topological placement of the synapses, along with neuron numbers and types, are all under GA control. Self-adaptive mutation provides the GA with as much freedom as possible by allowing specific subcomponents of each network to mutate at varying rates.

Despite numerous recorded successes when spiking networks are used to control ground-based robots (e.g., (Floreano et al., 2003a; Rocke et al., 2007)), to date no spiking neuro-evolution has been carried out on highly dynamic flying platforms. We wish to investigate the benefits of pairing a dynamic, stateful controller representation with a problem that displays the same properties. Our hypothesis is that the properties of the spiking networks make them especially suited to the control of dynamic agents in dynamic scenarios. To test this hypothesis we incrementally evolve a population of spiking controllers in increasingly challenging wind conditions and compare performance to a hand-tuned PID controller and evolved Multi-Layer Perceptron (MLP) neural networks (Rumelhart and McClelland, 1986). The controllers must keep the quadrotor as close to a pre-defined waypoint as possible, in spite of prevaling wind conditions.

The main contribution of this paper is the first use of evolutionary computing to generate spiking quadrotor neuro-controllers. We additionally provide evidence that even simple plasticity schemes can be a useful provider of adaptation in dynamic scenarios, and introduce a novel action encoding scheme that permits the spiking networks to operate at a higher frequency than those using traditional e.g., rate-based encoding schemes.

Background

Spiking Neuro-Controllers

Spiking neural networks are a bioinspired information processing paradigm based on studies of neural structure and activity in the brain. A number of neurons are linked via unidirectional, weighted connections. Each neuron has a measure of excitation (membrane potential) and communicates to other neurons via voltage spikes. A neuron spikes when its membrane potential exceeds some threshold, which typically requires a cluster of spikes arriving at the neuron within a short time period. Produced spikes are received by all forward-connected neurons. As membrane potential provides implicit memory, spiking networks are able to produce temporally dynamic activation patterns (Maass, 1997). Saggie-Wexler et al. (2006) highlight that this property makes them particularly suited for temporal problems such as robot control as steady-state errors and unmodelled dynamics can be accounted for. Recent studies in cognitive science highlight the importance of dynamic processes in memory and learning (Buzsaki, 2006).

Non-evolutionary approaches to spiking robot control include dynamic learning of navigation behaviour via conditioning (Helgadottir et al., 2013) and real-time training of neuro-controllers based on resovoir computing (Burgsteiner, 2005). Wiles et al. (2010) creates biologically plausible models of a rat's brain via self-organised learning for phototaxis. Commonalities between the studies include reports of adaptive learning, high performance in dynamic environments, and fast-changing nonlinear behaviour generation.

Evolutionary Robotics (Nolfi and Floreano, 2001) involves the use of evolutionary algorithms to design robot controllers. Neural networks are frequently used in such a setting as they are especially amenable to evolutionary search for beneficial compositions of connection weights and network topology — generally termed neuro-evolution - and have been shown to generate high-performance controllers. Hagras and Sobh (2002) evolve spiking networks for online robot control. Full topology-plus-weight evolution is used for both monolithic (Clune et al., 2009) and ensemble (Howard et al., 2010) controllers. Of particular interest is the work of Dario Floreano's group, who produce compact controllers for ground-based robots (e.g., Floreano et al. (2003a)), and evolve a spiking controller for an autonomous blimp (Floreano et al., 2003b). They demonstrate that even simple spiking representations can effectively process temporal state information.

Hebbian plasticity (Hebb, 1949) is thought to provide the brain with the properties of adaptation and self-organisation. Mechanistically, if one neuron can be said to have caused the subsequent firing of another neuron within a short time window, the connection strength between the two neurons increases and more strongly correlates their future activity. The reverse rule causes weight decay if the postsynaptic neuron fires first. Plasticity can provide unsuper-

vised in-trial learning, and as such is a valuable inclusion in robotic control systems that can be directly implemented in most spiking network models. Previous research by Wiles et al. (2010) demonstrates high performance of plastic networks in a frequency discrimination task in dynamic environments. Florian (2005) use a combination of STDP and global reward signals to effectively handle environmental uncertainty. Howard et al. (2014) show expedient adaptive re-organisation to recover from movement of the reward signal in the T-maze (Blynel and Floreano, 2003).

Evolving Rotorcraft Controllers

Evolution of rotorcraft controllers is difficult as finding an initial fitness gradient can be tricky — much of the network search space leads to controllers that crash the platform. Population seeding is often used to remedy this: three main examples include (i) incremental approaches (Hoffmann et al. (1998) generate fuzzy PID controllers for helicopter control), (ii) seeding with known successful controller parameters (De Nardi et al. (2006) evolve simple fixed-topology neurocontrollers that initially mimic PID control, and eventually evolve to outperform them; similar results are shown by Shepherd and Tumer (2010) for hierarchical waypoint-following tasks), and (iii) ensuring non-divergent networks in the initial population (Phillips et al. (1996) control a full-sized helicopter by evolving fuzzy modular controllers in a custom architecture based on how a human pilot flies a helicopter). All report higher performance when using seeding. Koppejan and Whiteson (2009) combine evolutionary algorithms and reinforcement learning for noisy hovering tasks similar to those we are interested in. Hovering tasks have been solved using Genetic Programming (Koza, 1992), e.g. by Dracopoulos and Effraimidis (2012).

Justification of our research methodology and approach is based on the knowledge that (i) spiking networks have previously been shown to be effective controllers for dynamic robotics tasks, (ii) evolved stateless neural networks (and other representations) have made promising candidate controllers for rotorcraft, and (iii) incremental approaches have shown to be beneficial given the difficulty of locating an initial fitness gradient. There is therefore some precedent to believe that incrementally evolved spiking quadrotor controllers will provide performance benefits when compared to stateless or traditional control schema in dynamic scenarios.

The System

To clarify the nomenclature that will be used herein: An *experiment* lasts for 500 evolutionary generations and focuses on a single controller in a single task. A *task* is a specific wind configuration in which the platform is tested. Each *generation*, 25 new controllers are created, trialled on the task, and assigned a fitness value. Each *trial* consists of a maximum of 300 timesteps and ends with a fitness value

being assigned. Each *timestep* begins with the current state being processed by the controller and ends with the resulting action being carried out and the simulator state updated. The state of the system is captured every 50 generations and used to generate the results. Fifteen experimental repeats for each controller type are used to create statistics and averages.

An incremental approach is used to focus the evolutionary search in areas that proved fruitful in similar, earlier tasks. This is designed to offset the poor initial fitness gradient caused by the catastrophic nature of failures, coupled with the fact that large portions of the network space were found to produce invalid controllers. For the first task, the population of networks is randomly generated with five hidden layer neurons. For subsequent tasks, the best 100 networks from the previous task are used as the initial population — each network is mutated at their current rates before being trialled on the problem.

Experimental Setup

We compare the ability of each controller type (spiking network, MLP, PID) to handle a series of four dynamic tasks. The QRSim (De Nardi, 2013) simulator was used as it models the physical platform we intend to eventually deploy the controllers on, and provides a high-fidelity wind model.

The quadrotor is an Asctec Pelican, whose flight performance is captured through system identification. Sensors and motors are subject to realistic normally-distributed noise; pertinently the GPS is modelled in detail accounting for factors such as varying numbers of available satellites and errors associated with transmission through various atmospheric layers. The quadrotor's software layer — which transforms desired control inputs into rotor RPM — is qualititively modelled to reflect the real platform.

All tasks take place in a three-dimensional arena with boundaries for each axis at $x_{min}/y_{min}/z_{min}$ =-10, $x_{max}/y_{max}/z_{max}$ =10 (z_{min} is ground level). At the start of a trial, the quadrotor in initialised at target waypoint W with position x=0,y=0,z=0 and the three control inputs are set to neutral (pitch θ =0°, roll ϕ =0°, thrust T=0.59) — see figure 1(a). Due to the complexity of the task, yaw is decoupled and stabilised to face North by a simple proportional controller. An image captured partway through a trial is shown in figure 1(b).

The quadrotor must stay as close as possible to W for 30 seconds of flight time — a control rate of 10Hz gives a maximum of t_{max} =300 timesteps per trial. At each timestep, the controller receives the noisy GPS-estimated difference between the quadrotor's estimated position and W in three dimensions, given as Δx_t , Δy_t , and Δz_t (units in metres, transformed to quadrotor body coordinates). Control inputs θ , ϕ , and T are calculated and passed to the simulator, which fuses them with the current yaw rate and wind state, and updates the platform state. Fitness f follows equation 1. Lower fitness indicates better hovering behaviour.

$$f = \sum_{t=0}^{t < t_{max}} \left(abs(\Delta x_t) + abs(\Delta y_t) + abs(\Delta z_t) \right)$$
 (1)

Violating a boundary causes trial termination and penalises the controller with a final fitness of $30\times$ the remaining number of timesteps in the trial, which is added to it's current fitness and is used as a pressure to generate controllers that can keep the quadrotor airborne.

We additionally compare the adaptation of the controllers to the task by recording the first generation that each experiment creates a *valid* controller G_{val} (that never violates a boundary), as well as the first generation with a controller that keeps the quadrotor within (i) 1.5m $(G_{1.5})$, (ii) 1m $(G_{1.0})$, and (iii) 0.5m $(G_{0.5})$ of the waypoint in every axis throughout the trial. If some of these criterea are not met by the end of the experiment, they are set to the maximum number of generations.

The tasks are as follows: (i) 1m/s steady wind, (ii) 1m/s gusting wind, (iii) 3m/s gusting wind, and (iv) 5m/s gusting wind, all with a general northerly direction. Wind is modelled as a noisy time-varying vector which is applied to the quadrotor centre of gravity. For steady wind, the magnitude of the vector is normally distributed around the mean values. Gusting wind follows a more in-depth model based on the Dryden spectrum, providing a plausible and challenging dynamic turbulence model — Liepmann (1952) provides full implementation details.

Spiking Controller

Leaky Integrate and Fire (LIF) (Gerstner and Kistler, 2002) networks are used as spiking controllers — three layers of neurons are connected by numerous weighted connections (weights constrained [0-1]), which can be recurrent within the hidden layer only. Neurons are either excitatory (transmit voltages $V \ge 0$) or inhibitory (V < 0).

On network creation, the hidden layer is populated with 5 hidden layer neurons, whose types are intitially excitatory with P=0.5, otherwise they are inhibitory. Each possible connection site is initially likely to have a connection with P=0.5, and all connections are weighted uniform-randomly in the range [0-1].

During activation, stimulation by incoming voltage alters the membrane potential y,y>0, which by default decreases over time. Surpassing a threshold y_{θ} causes a spike to be transmitted to all forward-connected neurons. The amount of voltage sent is equal to the weight of the connection, multiplied by -1 if sent from an inhibitory neuron. Immediately after spiking, the neuron resets its membrane potential to c. The membrane potential of a neuron at timestep t+1 is given in equation 2; equation 3 shows the reset formula.

$$y(t+1) = y(t) + (I + a - by(t))$$
 (2)

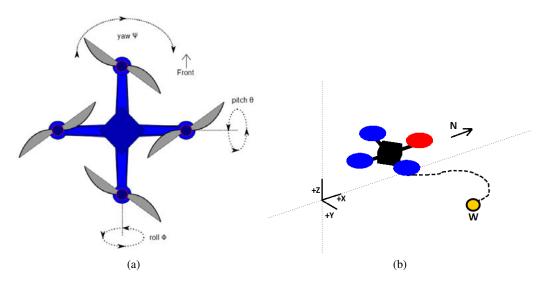


Figure 1: (a)Top-down view of the quadrotor showing pitch, roll and yaw. Thrust controls the homoegenous RPM of the rotor blades. Arrows show the direction of negative change.(b)The test environment, zoomed to show the quadrotor. The quadrotor (red rotor front) must stay as close as possible to the waypoint W (yellow circle). A sample path is shown (dashed line).

If
$$(y(t) > y_{\theta}) y(t) = c$$
 (3)

Here, y(t) is the neuron membrane potential at timestep t, I is the input voltage, a is a positive constant and b is the leak constant. A spike sent between two hidden layer neurons is received n (n > 1) steps after it is sent, where n is the number of neurons between the sending neuron and receiving neuron. A sample network is shown in figure 2. Parameters are a = 0.3, b = 0.05, c = 0.0, $y_{\theta} = 0.6$

A simple discrete-time linear model of plasticity is used to dynamically reconfigure the connection weights during the lifetime of the agent. The instantaneous weight w of a given connection is altered if it's two connected neurons spike in subsequent timesteps; no change occurs if both spike concurrently as causality cannot be implied. The sign of the weight change depends on which neuron — presynaptic (weight increase) or postsynaptic (weight decrease) — fires first, Δw =0.05 (experimentally determined). After a trial, each connection is reset to it's starting weight.

At a given timestep, three input neurons receive Δx_t , Δy_t , and Δz_t , which are scaled to the range [0,1]. The network sequentially updates it's input, hidden, and output layer neurons. Spike coincidences are checked and the plasticity rule applied to any affected synapses. Three pairs of output neurons control the commanded pitch θ (θ_{min} =-45°, θ_{max} =45°), roll ϕ ,(ϕ_{min} =-45°, ϕ_{max} =45°) and thrust T (T_{min} =0, T_{max} =1). If the first neuron spikes and the second does not, the relevant control variable is increased by 0.05 of it's total range (experimentally determined). If only the second neuron spikes, the control variable is decreased by the same amount. If neither or both neurons spike, the control variable is unchanged. Note that with this action en-

coding, (i) the total time to traverse the entire range of each variable is 4.4s, and (ii) the controller does not need to wait for multiple network processing steps before generating an action as with rate-based schemes. This will become more important in future experimentation with onboard control.

Comparitive Controllers We compare the spiking controllers (stateful, nonlinear) to PID controllers (stateful, linear) and MLP networks (stateless, nonlinear).

PIDs are chosen as they are a common provider of closed-loop control. The PID is hand-tuned experimentally from ground-truth readouts from the simulator. Full PIDs are used for attitude and altitude control. The provided PIDs have more stringent pre-defined operating limits of $\pm 20^{\circ}$ for θ and ϕ and, due to their linearity, do not operate well outside of these bounds. More of the flight envelope is therefore available to the spiking/MLP controllers $(\pm 45^{\circ})$.

MLPs are a challenging benchmark that have previously shown promising results for quadrotor control. Sigmoidal activation functions are used and each hidden layer neuron has a random bias between 0 and 1. Rather than mapping continuous outputs to continuous angles/thrust, a similar pairwise encoding to the spiking networks is used, i.e., increase in a given variable occurs when the output of the first neuron is >0.2 more than that of the second neuron. Initial experimentation (not shown) compared the chosen encoding to a standard continuous-output interpretation and indicated no significant differences between the two encodings, possibly due to the compression effect of the sigmoid function around midrange outputs. Plasticity is not used as no spiketime information is available.

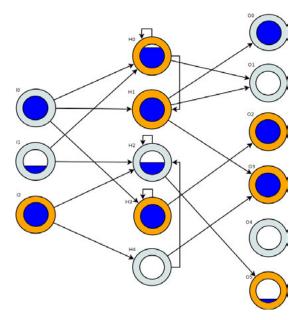


Figure 2: Showing a sample spiking network during activation. A neuron may be excitatory (orange) or inhibitory (grey), and has a membrane potential (centre circle). Membrane potential may build through a number of timesteps, eventually surpassing a threshold and emitting a spike (filled inner circle), before resetting (empty inner circle). Three input neurons take the differences in x, y, and z in the body frame between current position and target position. The six output neurons are paired; each pair controls either pitch, roll, or thrust. Actions are based on the presence or absense of instantaneous spikes at that timestep.

Genetic Algorithm

Per generation, following controller evaluation, the population is ranked based on fitness. The worst 25 networks are deleted. Children are then created up to the population limit of 100 by inverse-fitness-proportionate selection of a candidate parent, which is copied and probabilistically mutated before being inserted into the population. Crossover is omitted as — for these experiments — sufficient solution space exploration is obtained via a combination of weight and topology mutations; a view that is reinforced in the literature (e.g., by Rocha et al. (2003)).

Each network has it's own self-adaptive mutation rates, which are initially seeded uniform-randomly in the range [0,0.3] and mutated as with an Evolution Strategy (Rechenberg, 1973) as they are passed from parent to child following equation 4. This approach is adopted as it is envisaged that efficient search of weights and neurons will require different rates, e.g., adding a neuron is likely to impact a network more than changing a connection weight, so less neuron addition events than connection weight change events are likely to be desirable.

$$\mu \to \mu \ exp^{N(0,1)} \tag{4}$$

The genome of each network comprises a variable-length vector of connections and a variable-length vector of neurons, plus the mutation rates themselves. Different parameters govern the mutation rates of connection weights (μ) , connection addition/removal (τ) , neuron addition/removal (ω) , and neuron type change (ζ) . For each comparison to one of these rates a uniform-random number is generated; if it is lower than the rate, the variable is said to be satisfied at that allelle. During GA application, for each connection, satisfaction of μ alters the weight by ± 0 -0.1. Each possible connection site in the network is traversed and, on satisfaction of τ , either a new connection is added if the site is vacant, or the pre-existing connection at that site is removed. ω is checked once, and equiprobably adds or removes a neuron from the hidden layer if satisfied. ζ is checked once per neuron, and changes a neuron's type from excitatory to inhibitory (or vice versa) if satisfied. New neurons are randomly assigned a type, and each connection site on a new neuron has a 50% chance of having a connection. New connections are randomly weighted between 0 and 1.

Results

In the following comparisons, statistical significance is assessed using twin-tailed t-tests. Results are significantly different when p<0.05. Initial experimentation on the first task justifies the use of plasticity. Compared to identically-evolved spiking networks with static connection weights, plastic networks have significantly better average best fitness (50.32 for plastic networks vs 55.21 for non-plastic), as well as faster creation of controllers that attain $G_{1.5}$ and $G_{1.0}$ (average 6.7 vs 18.3 generations), and $G_{0.5}$ (average 9.3 vs. 24.5 generations).

Performance

Figure 3(a) shows the performance of each controller through all experiments. We note that MLP networks find the transition between environments to be more challenging than spiking networks (larger initial increases in fitness before recovery). In the final task, spiking networks start to pull away from the MLP and PID controllers. A gradual upwards trend in final fitness values reflects the increasing challenge of the tasks. Table 1 reveals that, for the first three tasks, both MLP and spiking controllers have significantly better final fitness than PIDs. For the final task, spiking controllers have significantly better final fitness than MLPs and PIDs. Average fitness values are strongly influenced by the number of valid controllers produced per generation, and reflects the ability of the GA to create child networks that inherit useful features from highly-fit parent networks. This ability may be useful for future on-line/on-board evolution. Spiking controllers are found to be more "evolvable" in this

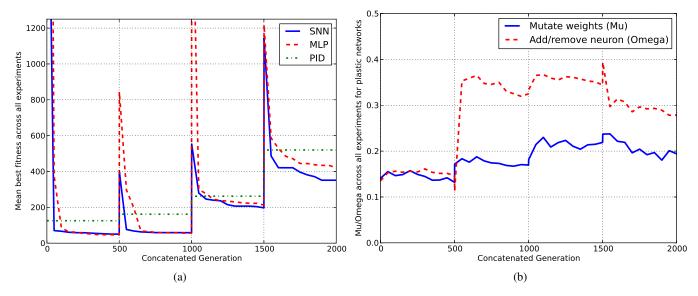


Figure 3: (a) Fitness recovery between tasks. MLPs suffer from greater initial fitness spikes and take longer to recover. (b) Mutation rates for connection weights and frequency of node addition/removal events for spiking networks across tasks. Note (i) the significant variance of final used mutation rates for each task, (ii) the gradual increase in weight mutation and neuron addition/removal events in response to more challenging environments. MLPs display similar trends.

sense — for the final two tasks spiking networks have significantly better average fitness than MLPs. Related to this, spiking networks were found to be more quickly able to evolve controllers that displayed the ability to hover. For all tasks, spiking networks were significantly faster than MLPs at attaining $G_{1.5}$, and for the first two tasks at attaining $G_{1.0}$ and $G_{0.5}$. Only spiking networks achieve $G_{0.5}$ in task 3, and no controller type achieves $G_{0.5}$ in task 4. Performance differences in all cases are attributed to the ability of the spiking networks to handle temporal phenomena (all other experimental parameters were identical, network action encodings were made as similar as possible).

We additionally note that in tasks 3 and 4, spiking and MLP networks utilised their extended flight envelopes to quickly recover from being gusted; the highest angle used was for spiking networks in task 4 (34°), compared to the PID maximum of 20°. This behaviour was seen more frequently in the later tasks where stronger responses were required. Due to the amount of noise present and the variability of wind gusts, general controllers — unable to specialise to the conditions — were evolved.

Network Composition

Both neural networks benefit from the compressive effect of connection disabling; the best network of each type per task was always <50% connected. In the first task, MLP networks required significantly fewer connections and neurons than spiking networks (average 44.2% / 12.4 vs 48.3% / 14.8 respectively). The opposite is true in the final task (MLP av-

erage 50.9% / 15.2, spiking average 46.1% / 10.1). Smaller networks reduce computational requirements, increasing the chances that prospective onboard implementation will be viable. Plasticity increases with task difficulty, with significantly higher mean numbers of weight change events per network between tasks 1 and 3, 1 and 4, and 2 and 4, indicating that the extra flexibility permitted by the plasticity scheme is harnessed by the network.

Mutation Rates

Mutation rates generally increase throughout to allow sufficient exploration in more challenging tasks. Larger jumps are evidenced at the start of a new task (figure 3(b)) as the controllers require additional search of the network space to tailor the current population to the more complex task. After these peaks, small downward trends are evidenced as stability is brought into the search process. Peak size indicates how difficult the transition is for the controller.

Final rate values vary across tasks for the same controller. This highlights the context-sensitivity of the process and leads us to believe that a single mutation rate per parameter would be suboptimal overall. μ (rate of connection weight change), ω (rate of neuron addition/removal), and τ (rate of connection enabling/disabling) are often significantly higher in spiking networks than MLP networks — see table 2. A possible explanation is that the network types fundamentally differ in how they perform input-output mappings, and the spiking network space requires more variable search to locate fit solutions as a result.

Task	Controller	Best fit	Avg fit	G_{val}	$G_{0.5}$	$G_{1.0}$	$G_{1.5}$
	Spiking	50.3 (4.3)*	497.6 (127)	3.8 (3.9)†	9.3(5.2)†	6.7 (4.8)†	6.7 (4.8)†
0.5m/s	MLP	46.4 (6.3)*	465.9 (141)	11.3 (5)	65.9(17.5)	48.5 (17.1)	42.7 (16.3)
	PID	125.2 (55.5)	NA	NA	NA	NA	NA
	Spiking	58.6 (7.5)*	352.3 (120)	0 (0)	5.6 (4.0)†	0.5 (0.98)†	0 (0)†
0.5m/s	MLP	56.1 (8.6)*	438.4 (82)	0 (0)	75.8 (34.8)	56.2 (34.2)	10.1 (8.9)
Gust	PID	161.5 (75.1)	NA	NA	NA	NA	NA
	Spiking	197.1 (24.3)*	811.4 (82) †	0 (0)	499.3 (2)	62.2 (4.8)	5 (6.7)†
3m/s	MLP	212.6 (34.4)*	1258.5 (108)	0.3 (0.7)	500 (0)	48.5 (17.1)	18.9 (16.1)
Gust	PID	262 (88.5)	NA	NA	NA	NA	NA
	Spiking	346.7 (68.4) † *	1555 (78) †	0 (0)	500 (0)	451.6 (102)	185 (174) †
5m/s	MLP	421.26 (54.8)	1746 (80)	0 (0)	500 (0)	500 (0)	391.2 (148)
Gust	PID	520.1 (179)	NA	NA	NA	NA	NA

Table 1: Averages and standard deviations for performance parameters of Spiking, MLP, and PID controllers across all four environments. Symbols indicate the controller type is statistically (p<0.05) better than \dagger = MLP, * = PID.

Task	Rate	Spiking avg (s.d)	MLP avg (s.d)
1	μ	0.132 (0.08)	0.024 (0.008)
	τ	0.017 (0.003)	0.012 (0.001)
2	μ	0.17 (0.04)	0.052 (0.03)
	ω	0.325 (0.12)	0.173 (0.137)
	au	0.103 (0.07)	0.012 (0.002)
3	ω	0.344 (0.12)	0.185 (0.07)
	au	0.161 (0.05)	0.024 (0.009)
4	τ	0.3 (0.09)	0.054 (0.01)

Table 2: Detailing averages and standard deviations for significantly different final self-adaptive mutation rates between the network types.

Effect of Incremental Evolution

In separate tests (not shown) we evolved spiking and MLP controllers for task 4 with random initial populations — neither produced a valid controller. This confirms that incremental evolution is seeding subsequent populations in useful areas of the search space. We also note that values of G_{val} for both networks generally decreases between experiments, i.e. valid controllers are evolved quickly despite the increasing difficulty of the task.

Discussion

We have demonstrated the benefits of spiking controllers for the control of dynamic platforms for dynamic tasks. The spiking controllers provided performance benefits on all tasks considered, with the fitness gap between spiking and MLP/PID control increasing with increasing task difficulty. Incremental evolution was found to be beneficial — seeded populations in harder tasks were significantly faster at creating valid controllers than random populations in the original, easiest task. Self-adaptive genetic search was found to

be beneficial in tailoring search rates to the task considered by allowing the various constituent parts of the network to vary at different rates, and by allowing increased network space exploration at the start of each new task to allow the controller to adapt to the environment.

Referring back to our hypothesis, we have shown that spiking networks were more able to cope with the unpredictabilities inherent in the tasks than the comparative controllers, possibly through harnessing latent temporal information in the wind which was effecting the quadrotor. Spiking controllers were significantly more rapid than MLPs in keeping the quadrotor within at least one of the predefined bounds $(G_{0.5}/G_{1.0}/G_{1.5})$ on all tasks considered.

Results encourage further experimentation — we aim to transfer our controllers to a physical quadrotor and expect that plasticity will play a more prominent role in providing online adaptation to discrepances between simulation and reality. We also wish to investigate the possibility of harnessing additional bio-inspired spiking mechanisms and integrating them into the neuro-controllers.

Acknowledgements

This work was funded as part of the OCE Evolutionary Aerial Robotics project.

References

Blynel, J. and Floreano, D. (2003). Exploring the t-maze: Evolving learning-like robot behaviors using ctrnns. In Cagnoni, S., Johnson, C., Cardalda, J., Marchiori, E., Corne, D., Meyer, J.-A., Gottlieb, J., Middendorf, M., Guillot, A., Raidl, G., and Hart, E., editors, *Applications of Evolutionary Computing*, volume 2611 of *Lecture Notes in Computer Science*, pages 173–176. Springer Berlin / Heidelberg.

Burgsteiner, H. (2005). Training networks of biological realistic spiking neurons for real-time robot control. In *Proceedings of the 9th international conference on engineering applications of neural networks, Lille, France*, pages 129–136.

- Buzsaki, G. (2006). Rhythms of the Brain. Oxford University Press.
- Clune, J., Beckmann, B. E., Ofria, C., and Pennock, R. T. (2009). Evolving coordinated quadruped gaits with the hyperneat generative encoding. In *Evolutionary Computation*, 2009. *CEC'09. IEEE Congress on*, pages 2764–2771. IEEE.
- De Nardi, R. (2013). The qrsim quadrotors simulator. *RN*, 13(08):08.
- De Nardi, R., Togelius, J., Holland, O., and Lucas, S. (2006). Evolution of neural networks for helicopter control: Why modularity matters. In *Evolutionary Computation*, 2006. CEC 2006. IEEE Congress on, pages 1799–1806.
- Dracopoulos, D. C. and Effraimidis, D. (2012). Genetic programming for generalised helicopter hovering control. In *Proceedings of the 15th European conference on Genetic Programming*, EuroGP'12, pages 25–36, Berlin, Heidelberg. Springer-Verlag.
- Floreano, D., Schoeni, N., Caprari, G., and Blynel, J. (2003a). Evolutionary bits'n'spikes. In *Proceedings of the eighth international conference on Artificial life*, pages 335–344, Cambridge, MA, USA. MIT Press.
- Floreano, D., Zufferey, J.-C., and Mattiussi, C. (2003b). Evolving Spiking Neurons from Wheels to Wings. In *Dynamic Systems Approach for Embodiment and Sociality*, volume 6 of *Advanced Knowledge International, International Series on Advanced Intelligence*, pages 65–70. K. Murase and T. Asakura (eds.).
- Florian, R. V. (2005). A reinforcement learning algorithm for spiking neural networks. In *Symbolic and Numeric Algorithms for Scientific Computing*, 2005. SYNASC 2005. Seventh International Symposium on, pages 8–16. IEEE.
- Gerstner, W. and Kistler, W. M. (2002). Spiking Neuron Models: Single Neurons, Populations, Plasticity. Cambridge University Press.
- Hagras, H. and Sobh, T. (2002). Intelligent learning and control of autonomous robotic agents operating in unstructured environments. *Information Sciences*, 145(1):1–12.
- Hebb, D. O. (1949). *The organisation of behavior*. Wiley, New York.
- Helgadottir, L., Haenicke, J., Landgraf, T., Rojas, R., and Nawrot, M. (2013). Conditioned behavior in a robot controlled by a spiking neural network. In *Neural Engineering (NER)*, 2013 6th International IEEE/EMBS Conference on, pages 891– 894
- Hoffmann, F., Koo, T. J., and Shakernia, O. (1998). Evolutionary design of a helicopter autopilot. In in 3rd On-line World Conf. on Soft Computing (WSC3, pages 201–214.
- Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, Michigan.
- Howard, G., Bull, L., de Lacy Costello, B., Gale, E., and Adamatzky, A. (2014). Evolving spiking networks with variable resistive memories. *Evol. Comput.*, 22(1):79–103.
- Howard, G., Bull, L., and Lanzi, P.-L. (2010). A spiking neural representation for xcsf. In *Evolutionary Computation (CEC)*, 2010 IEEE Congress on, pages 1–8.

- Koppejan, R. and Whiteson, S. (2009). Neuroevolutionary reinforcement learning for generalized helicopter control. In Proceedings of the 11th Annual conference on Genetic and evolutionary computation, GECCO '09, pages 145–152, New York, NY, USA. ACM.
- Koza, J. R. (1992). Genetic Programming: vol. 1, On the programming of computers by means of natural selection, volume 1. MIT press.
- Liepmann, H. (1952). On the application of statistical concepts to the buffeting problem. *Journal of the Aeronautical Sciences* (*Institute of the Aeronautical Sciences*), 19(12):793–800.
- Maass, W. (1997). Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659 1671.
- Maass, W. and Zador, A. M. (1999). Dynamic stochastic synapses as computational units. *Neural Computation*, 11(4):903–917.
- Minorsky, N. (1922). Directional stability of automatically steered bodies. *Journal of the American Society for Naval Engineers*, 34(2):280–309.
- Nolfi, S. and Floreano, D. (2001). Evolutionary robotics. the biology, intelligence, an technology of self-organizing machines.
- Phillips, C., Karr, C. L., and Walker, G. (1996). Helicopter flight control with fuzzy logic and genetic algorithms. *Engineering Applications of Artificial Intelligence*, 9(2):175–184.
- Rechenberg, I. (1973). Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution. Frommann-Holzboog, Stuttgart, Germany.
- Rocha, M., Cortez, P., and Neves, J. (2003). Evolutionary neural network learning. In *Progress in Artificial Intelligence*, volume 2902 of *Lecture Notes in Computer Science*, pages 24–28. Springer Berlin / Heidelberg.
- Rocke, P., McGinley, B., Morgan, F., and Maher, J. (2007). Reconfigurable hardware evolution platform for a spiking neural network robotics controller. In Diniz, P., Marques, E., Bertels, K., Fernandes, M., and Cardoso, J., editors, *Reconfigurable Computing: Architectures, Tools and Applications*, volume 4419 of *Lecture Notes in Computer Science*, pages 373–378. Springer Berlin / Heidelberg.
- Rumelhart, D. and McClelland, J. (1986). *Parallel Distributed Processing*, volume 1 & 2. MIT Press, Cambridge, MA, USA.
- Saggie-Wexler, K., Keinan, A., and Ruppin, E. (2006). Neural processing of counting in evolved spiking and mcculloch-pitts agents. *Artificial Life*, 12(1):1–16.
- Shepherd, III, J. F. and Tumer, K. (2010). Robust neuro-control for a micro quadrotor. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 1131–1138, New York, NY, USA. ACM.
- Wiles, J., Ball, D., Heath, S., Nolan, C., and Stratton, P. (2010). Spike-time robotics: a rapid response circuit for a robot that seeks temporally varying stimuli. In 17th International Conference on Neural Information Processing (ICONIP).

Evolving Multimodal Behavior Through Subtask and Switch Neural Networks

Xun Li¹ and Risto Miikkulainen¹

¹The University of Texas at Austin xun.bhsfer@utexas.edu

Abstract

While neuroevolution has been used successfully to discover effective control policies for intelligent agents, it has been difficult to evolve behavior that is multimodal, i.e. consists of distinctly different behaviors in different situations. This article proposes a new method, Modular NeuroEvolution of Augmenting Topologies (ModNEAT), to meet this challenge. ModNEAT decomposes complex tasks into tractable subtasks and utilizes neuroevolution to learn each subtask. Switch networks are evolved with the subtask networks to arbitrate among them and thus combine separate subtask networks into a complete hierarchical policy. Further, the need for new subtask modules is detected automatically by monitoring fitness of the agent population. Experimental results in the machine learning game of OpenNERO showed that ModNEAT outperforms the non-modular rtNEAT in both agent fitness and training efficiency.

Introduction

In complex real-world tasks, intelligent agents need to exhibit multimodal behavior. For instance, in robotic soccer keepaway, keeper agents try to complete as many passes to each other as possible while a taker does its best to steal the ball. When a keeper agent controls the ball, it needs to locate its teammates and pass the ball safely. When a teammate controls the ball, it needs to move to open position and prepare to receive passes (Whiteson, Kohl, Miikkulainen and Stone, 2005). Similarly, in an OpenNERO battle, two teams of agents fight to eliminate the enemy team with their weapons. Agents in each team must locate and navigate to enemy agents when they hide behind obstacles, and attack enemy agents when they fall into the weapon's range (Stanley, Bryant, Karpov and Miikkulainen, 2006).

To produce effective multimodal policies, complex tasks are often decomposed into tractable subtasks. Machine learning techniques can then be utilized to learn subtask behaviors. Subtask behaviors combined with an appropriate arbitration mechanism form a complete multimodal policy. The performance of such a policy depends on the division of the original task, the performance of subtask behaviors, and the mechanism for choosing among them. Therefore, an important research goal is to develop methods that decompose complex tasks automatically into tractable subtasks, learn effective behaviors in each subtask, and combine subtask behaviors to produce agents with effective multimodal behaviors.

A variety of methods exist for learning intelligent behaviors based on policy search; one of the most promising such methods is neuroevolution. Neuroevolution has been shown effective in discovering single control behaviors and has been widely applied to various control problems such as multilegged walking (Valsalam, Hiller, MacCurdy, Lipson and Miikkulainen, 2012; Clune, Beckmann, Ofria and Pennock, 2009), automated driving (Kohl, Stanley, Miikkulainen, Samples, and Sherony, 2006; Togelius, Lucas, and Nardi, 2007) and finless rocket control (Gomez and Miikkulainen, 2003). However, it is a challenging task to learn multimodal behaviors through neuroevolution.

One approach is to implement module mutation as part of the algorithm. This approach includes a structural mutation operator that adds a new module to a neural-network-based policy (Schrum and Miikkulainen, 2009). Several methods to implement module mutation have been proposed including Module Mutation Previous (MMP), Module Mutation Random (MMR), and Module Mutation Duplicate (MMD) (Schrum, 2014). Another approach is to use indirect encoding such as HyperNEAT to create the modules. Experimental results indicated that HyperNEAT is able to produce modular solutions rapidly for simple tasks that require multimodal behaviors (Clune, Beckmann, McKinley and Ofria, 2010). Various indirect encoding approaches can also be used to incorporate multimodality better in HyperNEAT (Pugh and Stanley, 2013). On the other hand, hierarchical learning approaches such as concurrent layered learning and coevolution were shown to work well in robotic keepaway soccer using neuroevolution with Enforced Sub-Populations (ESP). However, in the full robocup soccer simulator, homogeneous agents controlled by monolithic networks performed the best (Subramoney, 2012).

The prior studies concentrated on structural modification of neural-network-based policies, i.e. the initial structure and weights of new modules and how these modules could be best combined with existing modules. Thus, they did not address the problem of how such drastic modifications should be timed. Moreover, they did not demonstrate how proper arbitration between the subtasks could be discovered while evolving the subtask networks. This article proposes Modular NeuroEvolution of Augmenting Topologies (ModNEAT) as a solution to these problems. ModNEAT detects the need for module mutation by monitoring population fitness. Experimental results in the machine learning game of OpenNERO showed that ModNEAT is effective in finding appropriate task division and learning multimodal behaviors for complex tasks.

The remaining sections of this article are organized as follows: the next section introduces the ModNEAT algorithm, the third section presents and analyzes experimental results based on OpenNERO, and the fourth section reviews key ideas of this article and points out directions for future work. Following that is a brief conclusion.

Method

Modular NeuroEvolution with Augmenting Topologies (ModNEAT) is an evolutionary computation algorithm for learning multimodal policies with minimum human guidance. In this section, task decomposition and policy structure in ModNEAT are first introduced, followed by a description of the NEAT algorithm used for evolving the switch network and the subtask networks in a multimodal policy. The mechanism for adjusting the probability of module mutation based on population fitness is introduced next, and evolution of subtask and switch networks is discussed in the end.

Task Decomposition

To learn multimodal behaviors effectively, complex tasks are often decomposed into multiple subtasks (Jain, Subramoney and Miikkulainen, 2012). Machine learning techniques are applied to each subtask, and subtask policies combined to form a complete multimodal policy. In a neural-network-based multimodal policy, subtask policies are represented by different neural networks. Various mechanisms including human-specified rules, decision trees, and switch networks can be used to arbitrate among subtask policies (Whiteson, Kohl, Miikkulainen and Stone, 2005).

This approach is beneficial for two reasons. First, neural networks that are competent in one subtask may need to be adapted significantly to perform well in another subtask. By training neural networks separately for each subtask, selection in evolution becomes more focused, improving

training efficiency and resulting in better agent behaviors. Second, task decomposition allows agents to learn complicated behaviors by combining behaviors from different subtasks. Consequently, agents can achieve high fitness in a variety of complex situations.

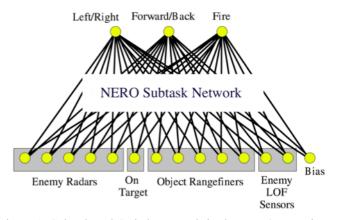
Whereas subtasks are usually defined based on human knowledge, ModNEAT performs task decomposition automatically. The method to detect the need for initiating new subtasks is introduced in the third subsection.

To allow greater flexibility for evolution, ModNEAT uses switch networks to combine subtask solutions (Figure 1). The complete policy of an agent contains: (1) n subtask networks that receive identical inputs, and (2) a switch network that receives the same inputs as the subtask networks and arbitrates among subtask networks, i.e. to decide which subtask network takes control of the agent during the current time step.

Since both switch networks and subtask solutions are neural networks, all components of a multimodal policy can be evolved through neuroevolution. In ModNEAT, fitness is defined based on the performance of the complete agent. The genetic encoding of the switch network and one or more subtask networks are concatenated into a single genome and evolved using the NEAT neuroevolution method as will be described next.

NeuroEvolution of Augmenting Topologies (NEAT)

Neuroevolution of Augmenting Topologies (NEAT) is a genetic algorithm for evolving artificial neural networks (Stanley and Miikkulainen, 2002). The algorithm for starts with an initial population of random networks. A fitness function based on domain knowledge is used to evaluate the neural networks in all generations. Those with the highest fitness are selected as parents of the next generation. Through mutation and crossover among the parents, a new generation of networks is created, and the next iteration of evolution begins. This process repeats until a satisfactory solution is found.



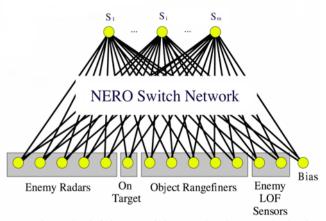


Figure 1: Subtask and Switch Network in the NERO Domain – The network on the left is one of the m subtask networks in the complete solution, and the network on the right is the switch network. Nodes in the bottom represent inputs from the environment, and S_1 to S_m are the outputs of the switch network. Each output of the switch network specifies the preference, for each of the m subtask networks, given current inputs. The output of the subtask network with the maximum weight is chosen as the output of the whole system. In this manner, the switch network selects the appropriate subtask network for each input situation.

In NEAT, mutation and crossover are not limited to modifying the weights of the neural networks. Topological augmentation is allowed so that various topologies can evolve, providing more flexibility for the genetic algorithm. Neural networks in NEAT are adapted in four ways:

- 1. Weights of one or multiple links are modified.
- 2. Weights are crossed over.
- 3. A link is added between two existing nodes.
- 4. A node is inserted in the middle of an existing link.

Note that modifications 3 and 4 are both topological augmentations. They allow the algorithm to start with relatively simple initial networks and gradually increase their topological complexity.

To protect innovation, when a new link is added between two existing nodes, the initial weight of the new link is close to zero. Similarly, when a new node is added, the in-link to the new node has a weight that is close to one, and the outlink from the new node preserves the weight value of the original link. Consequently, the initial impact of topological augmentation is limited, and networks with new structures are more likely survive in the evolution process.

In addition, to guarantee reasonable crossover between genotypes, an innovation number (global ID) is assigned for each new structure. For instance, when a new node is inserted to an existing link, the ID of the original link becomes obsolete, and a new ID is assigned to the each of the in-link, out-link, and the new node. Each genotype maintains a list of IDs for its structures (nodes and links), and crossover is limited to structures with identical IDs.

In order to protect innovation, NEAT speciates the population based on genotype similarity, which is a weighted sum of similarity score of weighting parameter values and network topological structures. Network fitness is compared primarily within species so that networks with innovations are offered enough time for optimization. Explicit fitness sharing (Goldberg and Richardson, 1987) is used for fitness evaluation so that species size is limited and no single species can easily take over the entire population.

In game domains, generation changes could result in noticeable discontinuations. Therefore, in the rtNEAT version of NEAT (Stanley, Bryant, and Miikkulainen, 2005), agents evolve while maintaining constant interaction with task environment. Instead of updating generations of populations, rtNEAT generates offspring continuously, one at a time. Thus, evolution is gradual and less visible to the human observers. Since ModNEAT experiments were run in the OpenNERO games, they were based on the rtNEAT method as well.

Automatic Subtask Initiation

One key issue in task decomposition is to detect the need for subtask initiation, i.e. to figure out when it is necessary to insert the appropriate structure (i.e. subtask networks and switch networks) to the current policy so that a new subtask can be initiated.

In ModNEAT, the need for subtask initiation is detected automatically by monitoring population fitness in consecutive generations. Intuitively, when the standard deviation of agent fitness remains low in multiple consecutive generations (i.e. evolution is not able to produce significant variation any longer), additional subtask networks

may be needed. The probability of subtask initiation in ModNEAT is defined as:

$$p_{s} = \begin{cases} 0, & n_{i} < N \\ \frac{(n_{i} - N)}{N}, & N \le n_{i} < 2N \\ 1, & n_{i} \ge 2N, \end{cases}$$
 (1)

where p_s is the probability of initiating subtask networks, n_i is the number of consecutive *ineffective generations* after the ith iteration of evolution, and N is an impatience parameter. When the evolution process is effective, p_s remains zero; when evolution falls into a performance plateau, the probability of subtask initiation increases linearly to one. Ineffective generation is defined as one where the standard deviation of agent fitness is less than a threshold parameter μ . Note that while multiple generations may be ineffective during evolution process, only consecutive ineffective generations may result in higher probability of subtask initiation.

Solution Evolution

ModNEAT starts with a population of simple initial policies. These policies can be either randomly generated or human-specified. Each initial policy contains a single subtask network and an initial switch network. The switch network contains only one output and grows as more subtask networks are added to the policy.

When a new subtask is initiated, a randomly initialized subtask network is added to the policy, and a new output node is added to the switch network. Three measures are taken to protect innovations:

1. Solution Complexity Control

When a new subtask network is added to a genotype, no other subtask networks can be added to the same genotype in the next N (the impatience parameter in equation 1) generations of evolution. Solution Complexity Control prevents unnecessary insertion of subtask networks and limits the complexity of the complete solution.

2. Prior Feature Protection

All existing subtask networks and switch network structures of a genotype are frozen in the next N iterations of evolution if a new subtask network is added to the genotype. In this manner, existing features that are advantageous in other subtasks are protected.

3. Population Speciation

As is mentioned in the previous subsection, population speciation is used to protect innovations. Since similarity scores between genotypes with different number of action subtask networks tend to be low, genotypes that contain additional subtask networks are usually regarded as a new species and are protected by speciation, which provides more time for new subtask networks to evolve.

Throughout the evolution process, fitness is evaluated for the policy as a whole. All modules, i.e. subtask networks and the switch network, are encoded in a single concatenated genome and therefore evolved together. Such evolution encourages subtask networks and switch networks that work well together to emerge. In addition, evolution combined with automatic subtask initiation encourages more complicated behaviors by combining existing subtask

behaviors to emerge. It is less likely for new subtask networks to be needed as the number of existing subtask networks increases. Thus, the overall complexity, i.e. the number of subtask networks in the multimodal policy, remains limited.

Experiments

This section presents experimental results based on the open source machine learning game OpenNERO to evaluate the effectiveness of ModNEAT. The first subsection introduces the OpenNERO platform, the second describes experimental setup, and the third compares ModNEAT with rtNEAT in both training efficiency and agent performance.

OpenNERO Platform

To test the effectiveness of ModNEAT, the performance of game agents generated by rtNEAT and Mod NEAT were compared in the OpenNERO implementation of the NERO machine learning game (opennero.googlecode.com; Karpov, Sheblak, and Miikkulainen, 2008; Stanley, Bryant, and Miikkulainen, 2005). In NERO, players design training schemes for a team of game agents. To defeat the enemy team, game agents must be able to target enemy agents and shoot them with their weapons. Since the arena contains obstacles such as walls and trees, agents need to be able to locate the enemy agents and navigate to them.

In the training mode of NERO, players set up training sessions by adding enemy turrets, flags, and walls. Also, game agents can be spawned at different locations, and all objects (flags, turrets, walls) can be moved around the arena. To evaluate fitness of the agents, players can set reward value for various behaviors such as sticking together, hitting enemy turrets, and getting closer to flags or enemy turrets. The overall fitness of an agent is a weighted sum of the rewards it obtains in each fitness measurement. A team of agents can be trained through a sequence of training sessions with different environment layouts and reward settings.

In the battle mode, two teams of agents are spawned on two sides of the middle wall in the arena. Each team consists of 50 game agents. Every agent has 20 hit points at the beginning. Whenever hit by enemy weapon, an agent loses one hit point. An agent is removed from the battlefield once it loses all hit points. The team that eliminates all enemy agents wins.

OpenNERO is an open-source version of NERO developed as a research and education platform. IT duplicates the original NERO game with a few minor differences on how the sensors and actions are implemented. In OpenNERO, every agent has 18 inputs excluding the bias:

- 1. Laser range finders that sense distance to the nearest obstacle (wall or a tree) in five directions: front, 45 degrees to each side, and 90 degrees to each side.
- 2. Radar sensors that return the distance to the flag in 5 overlapping sectors: -3 to 18, 12 to 90, -18 to 3, -12 to -90, and -90 to 90 degrees (behind the agent).
- 3. Radar sensors that trigger when enemies are detected within 5 sectors of the space (as defined above) around the robot. The more enemies are detected or the closer they

- are, the higher the value of the corresponding radar sensor will be.
- 4. Two sensors that depend on the distance and direction to the center of mass of the teammates.
- 5. A sensor that indicates whether the agent is facing an enemy within sensor range within 2 degrees.

The outputs of the control policy of an agent are:

- 1. Forward/backward speed: -1 to 1 of maximum. The agents can move at a rate of up to one distance unit per frame.
- 2. Turning speed: -1 to 1 of maximum. The agents can turn at a rate of up to 0.2 radians per frame.

Note that there is no output for taking shots. Instead, the agents shoot probabilistically. To attack an enemy, an agent must be oriented within 2 degrees of the target. Shooting accuracy increases linearly so that at 2 degrees an agent has a 50% chance of hitting, and if it face the center of the target exactly, it always hits. Also, the agent must be closer than 600 distance units from the target. Between 600 and 300, shooting likelihood increases linearly, and within 300, it always shoots.

OpenNERO provides a suitable platform to evaluate the performance of ModNEAT – the complex task of defeating the enemy team can be accomplished by combining behaviors in different subtasks such as navigating to enemy agents, aiming and shooting, and these behaviors can be learned sequentially through training sessions.

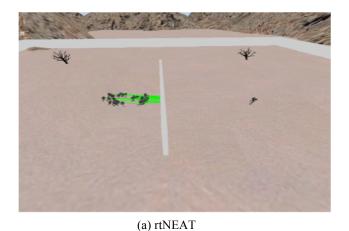
Experimental Setup

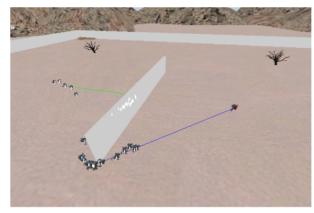
To compare the performance of ModNEAT with rtNEAT, two teams were trained and evaluated based on training time and tournament score. In order to ensure fair comparison, both teams were trained through identical training sessions. The objectives of each training session are listed in table 1. The training proceeded from S1 to S5, and a training session was terminated if average fitness of network population reached a preset value, or training time exceeded an hour.

In S1, a single enemy turret was placed in the arena, and there was no obstacle between spawning point and the turret. In S2, the enemy turret slowly moved around the spawning point to simulate moving enemy agents. In S3, an enemy turret was placed behind the wall (but far away from it), and agents were expected to go around the wall and hit it. In S4, enemy turrets were not only behind the wall but also moving around. In S5, enemy turrets were placed behind and close to the wall. Note that S5 is particularly difficult because agents must be able to tell if the enemy is in front of or behind the wall. If the enemy is behind the wall, agents should first go around the wall and then stop to shoot at the enemy.

Session	Objective
S1	Hitting still enemy in sight
S2	Hitting moving enemy in sight
S3	Hitting still enemy behind the wall
S4	Hitting moving enemy behind the wall
S5	Hitting enemy that hides closely behind the wall

Table 1: Training Sessions





(b) ModNEAT

Figure 2: Training Snapshots in S3 – Most rtNEAT agents (a) kept shooting at the wall. The green lines indicate that an obstacle blocks the shot. In contrast, most ModNEAT agents (b) managed to go around the wall using the new subtask network, i.e. subtask network 1. The white number on a ModNEAT agent is the ID of the subtask network that is currently active. They are using a different subtask network to get around the wall and shoot at the enemy.

In all of the above sessions, fitness reward was set to 100 for hitting enemy turrets. In S1, S2 and S5, rewards were 0 for all other possible behaviors (i.e. sticking together, approaching the flag or enemy turrets, etc), while in S3 and S4 reward for getting closer to enemy turret was set to 10 (all others were 0).

Results and Analysis

Table 2 shows the training time for each of the two teams. S3 is marked red because ModNEAT initiated most subtasks in this session, which significantly reduced the training time. The reason is that S3 requires significantly different network structures from S1 and S2, and it is difficult for rtNEAT to adapt networks that are trained for aiming and shooting at insight enemies to accomplish the task of reaching enemies behind the wall. As a matter of fact, in the first 25 minutes of S3 using rtNEAT, most of the agents simply kept shooting at the wall or running into it. In comparison, game agents

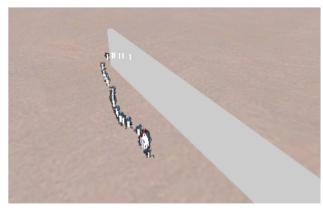
trained with ModNEAT managed to bypass the wall soon with the help of a new subtask network (Figure 2).

Session	rtNEAT	ModNEAT
S1	5	5
S2	5.5	6
S3	48.5	28.5
S4	7	8
S5	60+	12
Total	126+	59.5

Table 2: Training Time (minute)

Another important observation is that in session 5, the rtNEAT team never managed to reach satisfactory average fitness score while the ModNEAT team reached fitness goal within 12 minutes, even without introducing additional subtask networks. The reason can be found in the structure and function of the subtask networks.





(b) ModNEAT

Figure 3: Training Snapshots in S5 – Most rtNEAT agents (a) kept moving around the wall over and over again, i.e., they did not stay on the enemy's side to engage it. In contrast, ModNEAT agents (b) first navigated to the enemy turret using subtask network 1. When getting close, ModNEAT agents started attacking the enemy turret using subtask network 0. ModNEAT agents also learned to stay close to the enemy because they were more likely to hit the enemy turret when the distance is shorter.





Figure 4: Snapshots of a Battle between rtNEAT (red) and ModNEAT (blue) – This series of two snapshots shows how the ModNEAT agents exploit weakness of the rtNEAT agents. In the left picture, the two teams ran into each other from opposite directions. The rtNEAT agents continued marching around the wall after the encounter, but the ModNEAT agents start turning around to attack the enemies. The right snapshot is taken a few seconds later. After turning around, the ModNEAT team deals considerable damage to the rtNEAT team.

Team	ModNEAT	rtNEAT	me-rambo	synth.pop	synth-flag.pop	coward1
Max Score	649	542	512	409	398	359
Min Score	216	-649	-359	-542	-512	-471
Avg Score	424.1	253.5	187.5	82.09	68.45	13.18
Wins/Losses	11/0	10/1	8/3	8/3	7/4	6/5

Table 3: Agent Performance – Table 3 shows the performance of the ModNEAT team, the rtNEAT team, and the top four teams from the 2011 OpenNERO Tournament. The score of a team in a battle was the remaining hit points in the entire team minus that of the opponent team. Team A defeats team B in a match if the sum of A's two battle scores with B is positive. Wins/Losses indicates the number of matches a team won/lost.

For all ModNEAT agents whose final fitness was above the threshold, the complete solution consisted of two subtask networks. In session S1 to S4, subtask network 1 was used to get close to the enemy if it was far away. Subtask network 0 was then used to aim at the enemy and approach the enemy slowly while shooting. In session S5, when enemy turrets were placed right behind the wall, ModNEAT agents activated subtask network 1 to get around the wall. After getting close to the enemy, they switched to subtask network 0 to shoot the enemy. Thus, complex behaviors of reaching and eliminating well-hidden enemies were acquired by combining simple behaviors in different action modes.

In contrast, rtNEAT agents have only a single solution network which attempts to integrate both of the above functionalities. To reach enemy turrets right behind the wall, solution networks must learn to move around the wall. To deal enough damage to enemy turrets, solution networks must learn to aim at the enemy and focus on shooting. Consequently, rtNEAT evolved a compromise policy where agents keep moving around the wall and shoot enemy turrets on the way. Even though they eventually evolved to reach the enemy turrets, rtNEAT agents wasted too much time moving around the wall and therefore never managed to reach the fitness threshold for S5 (Figure 3).

In the complete solution of the ModNEAT agents, each subtask network contained 18 inputs and two outputs (as specified in the OpenNERO Platform subsection). Subtask network 0 of the champion agent in the last generation

contained six hidden nodes and 64 links, subtask network 1 contained eight hidden nodes and 72 links, and the switch network had four hidden nodes and 52 links. Note that most hidden nodes are not fully connected to the inputs and outputs, and all networks were non-recurrent. On the other hand, the solution network of the rtNEAT champion is a single network with 10 hidden nodes, 78 links, and the same number of inputs and outputs. Such network structures are typical for agents evolved by rtNEAT.

In order to evaluate the performance of the two teams, both teams were tested in a tournament against the top 10 teams in the 2011 OpenNERO Tournament (code.google.com/p/opennero/wiki/TournamentResults2011). Every team participated in 11 matches to compete with all 11 opponents. Each tournament match is consisted of two battles (with the spawning location switched). The results of the tournament are shown in Table 3.

While both of the two teams defeated all ten tournament teams, the ModNEAT team defeated the rtNEAT team by a large margin, i.e. by 649 hit points. The agents trained with rtNEAT (as well as the agents in many top-ranked teams in the 2011 OpenNERO Tournament) tended to keep moving around the wall over and over again after encountering the enemy. The ModNEAT agents exploited this weakness by switching to shooting mode while the enemy agents were busy moving ahead (Figure 4). Thus, the subtask and switch network architecture allowed evolving distinctively better behaviors than before.

Discussion and Future Work

Modular NeuroEvolution of Augmenting Topologies (Mod-NEAT) is an evolutionary computation method aimed at tackling complex tasks with minimum human guidance. The algorithm detects the need for adding additional subtask networks automatically by monitoring variation in population fitness. Switch networks are employed to arbitrate between subtask networks based on inputs from the environment. The subtask networks and switch networks are evolved through NEAT-based evolution together in a single genome. Experimental results showed that compared to rtNEAT, ModNEAT generates agents with significantly higher performance in shorter training time.

ModNEAT has higher training efficiency because it does not attempt to adapt networks trained for certain objectives to accomplish new objectives if they require substantially different network features (i.e. topological structure and link weights). Instead, new subtask networks are initiated and evolved to reach such objectives.

ModNEAT manages to learn more intelligent behaviors because task decomposition allows natural selection to be more focused on each subtask, and thereby makes subtask behaviors more robust and effective. In addition, ModNEAT has higher potential to learn intelligent behavior that is difficult to learn with a single network because switch networks allow relatively simple subtask behaviors to be combined flexibly to exhibit more complex behaviors.

An interesting direction of future work is to combine ModNEAT with various module-mutation techniques (Schrum, 2014) to develop more effective methods for initiating, evolving, and combining subtask networks. Another interesting idea is to explore other fitness-based mechanisms to adjust subtask initiation probability so that better task division can be found with fewer attempts. Third, ModNEAT can be applied to many challenging problem domains such as robot soccer and Non-player Character (NPC) design for video games to produce intelligent agents with effective multimodal behavior.

Conclusion

Complex real-world tasks often require intelligent agents to exhibit multimodal behavior. Although neuroevolution has been successfully used in learning single control behavior, it has been difficult to develop multimodal behavior in this approach. This article shows how such behavior can be constructed by discovering and evolving subtask networks with a switch network, i.e. using the ModNEAT approach. The results in the interactive game of OpenNERO show that ModNEAT is effective, i.e. it results in better policies and evolves faster than standard neuroevolution. Thus, ModNEAT can be useful in constructing intelligent agent behaviors for games and robotics in the future.

Acknowledgements

This research was supported in part by NSF grants DBI-0939454 and IIS-0915038, and in part by NIH grant R01-GM105042.

References

- Whiteson, S., Kohl, N., Miikkulainen, R., Stone, P. (2005). Evolving Keepaway Soccer Players through Task Decomposition. *Machine Learning*, 59(1):5–30.
- Stanley, K. O., Bryant, B. D., Karpov I., Miikkulainen, R. (2006). Real-Time Evolution of Neural Networks in the NERO Video Game. In Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-2006),1671–1674, Boston, MA.
- Valsalam. V. K., Hiller, J., MacCurdy, R., Lipson, H., Miikkulainen, R. (2012). Constructing Controllers for Physical Multilegged Robots using the ENSO Neuroevolution Approach. *Evolutionary Intelligence*, 5(1):1–12.
- Clune, J., Beckmann, B.E., Ofria, C., Pennock, R. T. (2009). Evolving Coordinated Quadruped Gaits with the HyperNEAT Generative Encoding. In *Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC)*, Trondheim, Norway.
- Kolh, N., Stanley, K. O., Karpov I., Miikkulainen, R., Samples, M., Sherony, R. (2006). Evolving Real-world Vehicle Warning System. In Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, 1681-1688, ACM, New York, NY, USA.
- Togelius, J., Lucas, S., and Nardi, R. (2007). Computational Intelligence in racing games. In: *Advanced Intelligent Paradigms in Computer Games*, pp. 39-69. Springer.
- Gomez, F. J., Miikkulainen, R. (2003). Active Guidance for a Finless Rocket Using Neuroevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2084-2095, San Francisco. MorganKaufmann.
- Schrum, J., Miikkulainen, R. (2009). Evolving Multimodal Behavior in NPCs. In *IEEE Symposium on Computational Intelligence and Games (CIG 2009)*, 325-332, Milan, Italy.
- Clune, J., Beckmann, B. E., McKinley, P. K., Ofria, C. (2010). Investigating Whether HyperNEAT Produces Modular Neural Networks. In *Proceedings of the Genetic and Evolutionary Computation* Conference, 635-642.
- Pugh, J. K., Stanley, K. O. (2013). Evolving Multimodal Controllers with HyperNEAT. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2013)*, 735-742, New York, NY: ACM.
- Subramoney, A. (2012). Evaluating Modular Neuroevolution in Robotic Keepaway Soccer. Master Thesis, Department of Computer Science, the University of Texas at Austin, Austin, TX.
- Jain, A., Subramoney, A., Miikkulainen, R. (2012). Task Decomposition with Neuroevolution in Extended Predator-Prey Domain. In Proceedings of Thirteenth International Conference on the Synthesis and Simulation of Living Systems, East Lansing, MI.
- Stanley K. O., Miikkulainen, R (2002). Evolving Neural Networks Through Augmenting Topologies. Evolution Computation, 10(2): 99-127.
- Goldberg, D. E., Richardson, J. (1987). Genetic Algorithms with Sharing for Multimodal Function Optimization. In *Proceedings of the Second International Conference on Genetic Algorithms*, page 148-154, Morgan Kaufmann, San Francisco, CA.
- Stanley, K. O., Bryant, B. D., Miikkulainen, R. (2005). Evolving Neural Network Agents in the NERO Video Game. In *Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games (CIG'05)*. Piscataway, NJ: IEEE.

- Karpov, I. V., Sheblak, J., Miikkulainen, R. (2008). OpenNERO: a Game Platform for AI Research and Education. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*.
- Schrum, J. (2014). Evolving Multimodal Behavior Through Modular Multiobjective Neuroevolution. PhD Thesis, Department of Computer Science, the University of Texas at Austin, Austin, TX.

POET: an evo-devo method to optimize the weights of large artificial neural networks

Alessandro Fontana¹, Andrea Soltoggio² and Borys Wróbel^{1,3}

¹Evolving Systems Laboratory, Faculty of Biology, Adam Mickiewicz University, Poznań, Poland
²Computer Science Department, Loughborough University, Loughborough LE11 3TU, UK
³Systems Modelling Laboratory, IO PAS, Sopot, Poland
fontana@evosys.org, a.soltoggio@lboro.ac.uk, wrobel@evosys.org

Abstract

Large search spaces as those of artificial neural networks are difficult to search with machine learning techniques. The large amount of parameters is the main challenge for search techniques that do not exploit correlations expressed as patterns in the parameter space. Evolutionary computation with indirect genotype-phenotype mapping was proposed as a possible solution, but current methods often fail when the space is fractured and presents irregularities. This study employs an evolutionary indirect encoding inspired by developmental biology. Cellular proliferations and deletions of variable size allow for the definition of both regular large areas and small detailed areas in the parameter space. The method is tested on the search of the weights of a neural network for the classification of the MNIST dataset. The results demonstrate that even large networks such as those required for image classification can be effectively automatically designed by the proposed evolutionary developmental method. The combination of real-world problems like vision and classification, evolution and development, endows the proposed method with aspects of particular relevance to artificial life.

Introduction

An important feature of artificial living systems is that of learning from and computing with rich sensory information. In biology, such feats are performed by large learning neural structures capable of developing and learning during lifetime. Computational models of large neural networks, however, are difficult to design due to the large search space. For example, artificial neural networks for image classification require computationally intensive learning algorithms to optimize large numbers of parameters (Bengio, 2009), and evolutionary computation was not, so far, an efficient way to do so (Koutník et al., 2013). This is especially true for evolutionary search with direct encodings, which ignore regularities, repetitions and patterns in the desired solutions. Indirect encodings (Lindenmayer, 1968; Stanley and Miikkulainen, 2003; Federici, 2004; Roggen and Federici, 2004; Hornby, 2005) were proposed as a more suitable representation to help evolutionary search. Such encodings are inspired by biology, where compact genotypes encode complex phenotypes such as the human body, a structure built of a very large number of interacting cells (in the order of 10^{13} ; Wolpert and Ticke, 2010; Bianconi et al., 2013).

Some models of direct encoding mimic biology by starting from a single element, from which a final structure grows, consisting of many such elements (Smith and Thelen, 1993; Bongard and Pfeifer, 2003; Kumar and Bentley, 2003; Roggen and Federici, 2004). Others, notably HyperNEAT (Stanley et al., 2009), produce a phenotype by means of a single-step indirect mapping function of a compact genotype. Indirect encodings, whether they model development or not, are characterized by compact representations of correlated parameters in the large phenotype space. Such compact encoding, however, biases the search, sometimes with detrimental effect, particularly in the presence of irregularities (Clune et al., 2009; van den Berg and Whiteson, 2013), even though these findings have been later put in question (Stanley et al., 2013). In short, while direct encodings are inefficient while searching regular patterns, indirect encodings suffer from the opposite problem of struggling with irregular and fractured search spaces.

The observation of the inefficiencies of both types of encoding inspired methods that can search both regularities as well as particularities in the solution space (van den Berg and Whiteson, 2013). The present study contributes to this line of research by demonstrating the capabilities of an evolutionary developmental method in searching large parameter space in a neural network for image classication. The proposed method is inspired by biological development, and based on gene expression mechanisms that can map genes locally with variable intensity, affecting at times large, at times small parts of the phenotype. For this reason, arbitrarily large search spaces can be searched without losing the ability to discover particularities.

The method proposed here introduces a novel parameter search technique that makes use of a biologically-inspired evolutionary developmental algorithm called *ET* (for *Epigenetic Tracking*). ET allows evolution and development of very large complex artificial systems built from cells with diverse cell types (Fontana, 2008), and modeling of biologically-relevant phenomena (Fontana, 2009), such as

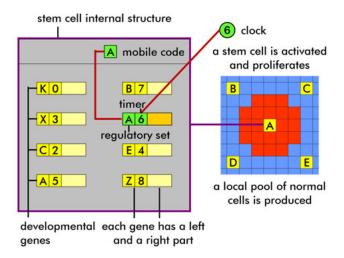


Figure 1: Structure of a stem cell. A stem cell is composed of one mobile code and several developmental genes. If the regulatory set of the developmental gene matches the mobile code of the cell and the timer matches the global clock, that particular developmental gene is activated and a change event (in this case a proliferation) is produced.

regeneration (Fontana and Wróbel, 2013a) and carcinogenesis (Fontana and Wróbel, 2013b) and the hypothetical transfer of genetic elements from soma to germline (Fontana and Wróbel, 2012). In the current study, cells are mapped to neural network weights and the method is tested on an image classification problem (hand-written images from the MNIST dataset; LeCun and Cortes, 1998). The results demonstrate, to the best of our knowledge, unmatched performance on the MNIST dataset with purely evolutionary methods.

The rest of the paper is organized as follows. We first describe ET as a model of developmental biology. We then explain the new method proposed here, and its application to searching weights in artificial neural networks. In *Experimental Results* we show how the method can be used in an image classification task. The final sections discuss the implications of our results and draw the conclusions.

The cellular model of development

This section gives an overview of ET, a model of development introduced in (Fontana, 2008), and belonging to the field of Artificial Embryology (Stanley and Miikkulainen, 2003). Notable examples of Artificial Embryology models are (Gruau et al., 1996; Eggenberger-Hotz, 1997; Cussat-Blanc, 2008). In ET, artificial bodies are composed of two categories of cells: *stem cells* and *normal cells*, placed on a grid. Artificial development starts with the value of a *global clock* set to 0, and with one or several stem cells on the grid. Each stem cell (Fig. 1) has a unique *mobile code*, but all

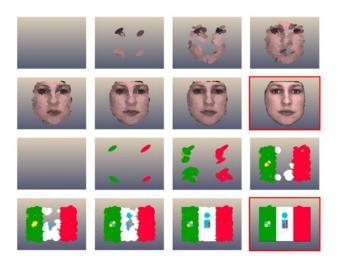


Figure 2: Illustration of the ability of ET to develop cellular structures similar to a target that contains regular patterns and irregularities. Two developmental sequences are shown, each in seven 1000×1000 frames (the frames with a red border are the targets).

cells have the same genome, which consists of *developmental genes*. All cells have access to the same temporal information provided by the global clock. As the clock advances, developmental genes are activated.

When a developmental gene is activated in a stem cell, the right part of the gene specifies a *change event* orchestrated by the cell. Two types of change events, *proliferation* and *apoptosis*, result—respectively—in filling a volume around the stem cell with new cells (displacing older cells if present), or deleting the cells around this stem cell (leaving an empty space). Where and when the change event occurs is specified by the left part of the developmental gene, consisting of a *regulatory set* and a *timer*. A gene is activated in the stem cell whose mobile code matches the regulatory set, and when the *global clock* matches the timer.

Thus, the mobile code corresponds in biology to regulators (such as transcription factors) specific for a given cell; the global clock corresponds to regulators that provide the temporal information in development (to which all cells have access); the regulatory set and the timer correspond to the regulatory sequences to which regulators can bind. In the current software implementation, the clock and the timer are integers, while the mobile code and the regulatory set are arrays of integers (each number in the array can be interpreted as a transcription factor or a regulatory locus, respectively).

After a proliferation, normal cells which are sufficiently distant (the distance is a parameter of the system) from any stem cell are turned into new stem cells. This process of stem cell formation is inspired by the recently emerging paradigm of dymanic stemness (Cruz et al., 2012; Roesch

et al., 2010). In this novel view, stemness is considered as a dynamic property: the stem-non stem conversion would occur in both directions, triggered by genetic and epigenetic factors, and influenced by the cellular microenvironment. In the ET implementation used in this paper, the stem cell formation mechanism results in stem cells evenly spaced in the body (a simplification of the biological reality introduced for computational reasons). Each new stem cell receives a new and unique mobile code.

This developmental model can be evolved by means of a *genetic algorithm*, and becomes an evo-devo process (examples of other evo-devo models are Joachimczak and Wróbel, 2009; Cheney et al., 2013). Briefly, in every generation a *fitness value* is obtained for each genome in the population. The fitness is computed by testing the network at the end of the developmental process, i.e. when the value of the global clock reaches a pre-specified value. The fitness may be determined as the proximity to a predefined target structure, as it was done in previous work (Fontana, 2008), or by other measures, as it is described in the next section.

A peculiar feature in this implementation of the genetic algorithm, called progressive freezing (Fontana, 2012), warrants a more detailed description. With progressive freezing, the genomes are separated in sections of G (20 in the current implementation) genes, each with the same timer value. Sections are executed in a sequential order, as the global clock advances. During simulated evolution, only one section of all genomes evolves for a number of generations, which means that mutations affect the regulatory sets and right parts only of the genes in the section being currently evolved. Development ends (and the fitness is calculated for the genome) when the clock strikes the number corresponding to the section under evolution. The other sections of the genome remain unchanged ("frozen"). A parameter specifies for how many generations a section evolves. More details of the genetic algorithm are provided in the Appendix.

From cell structures to neural networks

The coupling of the model of development and the genetic algorithm, described in the previous section, gives origin to an evo-devo process, which was proven capable to "devoevolve" structures of unprecedented complexity when proximity of the developed cellular structure to a pre-specified target was used as a fitness measure. Let us consider two 2-dimensional target structures that contain regularities and particularities (Fig. 2). Although the search of genomes regulating development of structures similar to a target is not the purpose of this study, these two examples show the capability of ET when dealing with search spaces with a large number of parameters (in this case, colored pixels of an image). In the present study, we build on this capability and propose an extension of ET that creates a more general method for parameter optimization. This new method is called POET (for Parameter Optimization using Epigenetic Tracking).

The following subsections describe how POET exploits 2-dimensional cellular structures to specify weights of a network with arbitrary predefined topology. In this initial study, the focus is limited to indirect developmental encoding of a large number of static weights. Once the cellular structure is interpreted as weights, they do not change during fitness calculation for a particular genome. The task considered is image classification. The fitness is computed on the performance of the network on the classification of sets of images. Using a machine learning terminology, the set of images used during evolution is called "training set". At the end of evolution, the evolved networks are tested on a set of images that was not seen during evolution. These additional images are the "test set".

Mapping the parameter space

Each weight of a neural network is associated with a progressive index $i \in \{0, N-1\}$ with N being the total number of weights. Each weight with index i is linked to k locations on the developmental grid—which may or may not be the location of a cell—by means of a mapping function $f: i \to \{(m, n)_1, (m, n)_2, ..., (m, n)_k\}, \text{ where } m, n \text{ are co-}$ ordinates on the grid (Fig. 3). How do cells contribute to a weight in the network? First, each cell has a real value in the range [-1, 1], specified by the genetic material. Then, the value of the weight i is derived by summing the k values of k cells specified by the mapping function f. The function f is initialized randomly and can be modified by a new change event called swap, orchestrated—in addition to proliferation and apoptosis, taken from ET-by dynamic stem cells during the growth of the cellular structure. Development starts from a number of stem cells initially placed on the grid (2500 evenly spaced stem cells in this study), and proceeds through a number of developmental stages (50 in our simulations). In each stage a maximum number of change events (20 proliferation, apoptosis and swap events in total) was allowed to take place.

Swap During a swap event, an area around the stem cell exchanges the values of the function f with another area of the grid (Fig. 4). The swap area has the shape of an ellipse whose size, elongation and orientation are specified in the right part of the associated gene. The centers of the two swapped areas are given by the location of the stem cell which gives origin to the swap event and by another couple of coordinates, also contained in the gene's right part. The swap event has the purpose to cluster in the same area of the grid parameters which are correlated, so that they can be optimized together by means of a proliferation or apoptosis event.

Proliferation and apoptosis As in standard ET, when a stem cell undergoes proliferation, the right part of the activated POET gene specifies the shape of the region filled by

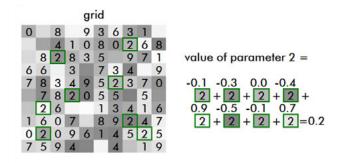


Figure 3: Parameter mapping in POET. To each cell two numbers are associated: a real valued number, represented by the intensity of shading, and an integer value (shown inside the cell). The value of parameter whose index value is i is calculated by adding all real valued numbers of cells whose integer value is i.

	grid										sw	ap	ev	ent	1				
0		8		9	3	6	3	1		0		8		9	3	6	3	1	
		4	1	0	8	0	2	6	8			4	1	0	8	0	2	6	8
	8	2	8	3	5		9	7	1		8	2	8	3	1	8	2	0	1
6	6		3		7	3	4		9	6	6		3	> (2	6			9
7	8	3	4	9	5	2	3	7	0	7	8	3	4	9	8	0	7	/	(
	7	8	2	0	5	5		5			5		9	7	5	5	1	5	
	2	6			1	3	4	1	6		7	3	4)∗	3	4	1	6
1	6	0	7		8	9	2	4	7	1	5	2	3	7	8	9	2	4	7
0	2	0	9	6	1	4	5	2	5	0	2	0	9	6	1	4	5	2	5
7	5	9	4			4		1	9	7	5	9	4			4		1	9
	before swap								af	er	sw	ар							

Figure 4: Swap event in POET. Two elliptical regions are selected in the grid and the values contained therein are swapped. This operation leaves the values associated with each cell in the grid (represented by the intensity of shading) unaffected.

new cells in the same way as the swap area, and the same goes for the area emptied during apoptosis (Fig. 5). The activated gene specifies also the differentiation state of the new cells resulting from proliferation. The state includes a real value that contributes to a weight in the neural network.

Neural network and classification

An input example in the MNIST set is a 28×28 pixel image. The resulting 784 inputs are fed to an equal number of input neurons in the input layer of the network. A hidden layer comprises 336 neurons, and the output layer has 80 neurons, divided in 4 groups of 20 neurons each (Fig. 6). The structure in this particular setting was devised to classify only the first four digits (0 to 3) of the MNIST dataset. In addition, the network structure is constrained such that the network may be seen as four separate networks with structure

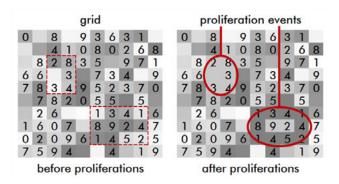


Figure 5: Proliferation events in POET. Proliferations events affect elliptical regions, changing the values (represented by the intensity of shading) inside the regions, without affecting the associated parameter indexes.

 $784 \cdot 84 \cdot 20$ for input, hidden and output neurons, respectively. Each network has $784 \cdot 84 + 84 \cdot 20 = 67536$ weights, resulting in a total of $67538 \cdot 4$ (subnetworks) = 270144 total weights. This large number is generally considered intractable for any evolutionary search method with direct encoding.

The constraints on the network topology imply that the algorithm is evolving effectively four separate networks for each class to be classified. The lack of processing units that extract common features to more classes is in contrast to other approaches in the literature, particularly deep learning. While the present method may indeed be applied to deep structures, the experiments presented here are a proof-of-concept of the potential of optimizing a very large parameter space.

Correspondingly, the grid was divided into four quadrants, each of which associated to a given subnetwork (Fig. 7). In other words, the weights of each subnetwork are mapped to one quadrant only, and the swap event is restricted in such a way as to preserve the "fencing" between quadrants. More specifically, the rule imposed states that, if the center of the source ellissoid of a swap event belongs to quadrant Q, also the center of the destination ellissoid is forced to belong to quadrant Q.

An image is assigned a class by observing which of the four output neuron groups has the highest activation value (the sum of activations of all members). In case the first and the second groups have the same activation value, the image is not assigned to any class. The fitness function is expressed as

$$Fitness = 0.75 \cdot (O_{cc} - O_h) + 0.25 \cdot sign(O_{cc} - O_h)$$

where O_{cc} is the average output of the neurons in the *correct* class, O_h is the average value of the highest output group excluding O_{cc} . The first term $(O_{cc} - O_h)$ is proportional to the margin between the values of the correct class and the best of

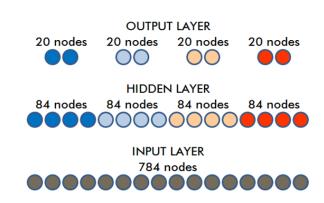


Figure 6: Structure of the neural network used for the classification of MNIST images (784-pixel images of digits). The network is composed of four subnetworks, each dedicated to a given class (one of four digits). Nodes of the output layer of subnetwork 1 (blue) are fully connected to nodes of the hidden layer of the same subnetwork, which are fully connected to all 784 input nodes. The same is true for the other subnetworks. There is no connections between neurons of different subnetworks.

the other groups (negative if the correct class is not correctly classified); the second term $sign(O_{cc}-O_h)$ introduces a nonlinear step which gives a fitness premium when a correct classification is achieved. This fitness measure creates a gradient in the fitness landscape towards a correct classification. Preliminary experiments (data not shown) indicated an advantage of this fitness function in comparison to simpler versions that considered only the number of correct classifications.

Experimental results

The experiments tackled the classification of four digits, from 0 to 3. Although only a part of the MNIST set was used in this first study, the results demonstrate the feasibility of purely evolved neural weights in image classification. The evolution was performed using 1000 training images for each digit. Fig. 8 shows examples of correctly classified images.

At the end of evolution, the set of evolved weights were extracted and tested on a different platform (MATLAB) to verify the performance both on the training set and on the test set of the MNIST dataset. The error rates in classification averaged over four simulations are summarized in Tables 1 and 2. Fig. 9 shows the developed organism of one of the champion networks. From this graphical representation, it is evident that the large search space makes it difficult to understand intuitively how the quarter of a million weights cooperate to perform classification. However, one hypothesis is that each cell proliferation contributes to the extraction

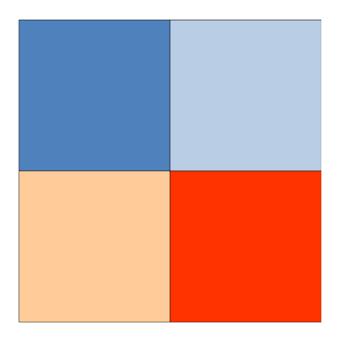


Figure 7: Subdivision of the grid into four quadrants. Each quadrant maps the weights associated to one of the four subnetworks which perform the classification task. The colors correspond to the colors in Fig. 6.



Figure 8: Examples of handwritten digits from the MNIST set that were correctly classified.

of useful features, which then contribute in the output layer to the correct classification.

The results outline two important points. The first is that a purely evolutionary algorithm was shown to solve image classification with a large neural network. Although the error rates are worse when compared to the state-of-the-art learning algorithms on the MNIST set, they nevertheless demonstrate the capability of the algorithm to perform classification in such a problem domain, which was traditionally considered intractable with evolutionary algorithms. In particular, it is possible to observe that the network acquires throughout evolution the capability of recognizing well written digits. Fig. 10A shows the case of the digit 3 that was recognized with the highest confidence, while Fig. 10B shows a digit 3 that was misclassified as 2. Interestingly, the second guess was 3. The examples indicate that the algorithm fails in a human-like fashion particularly when the hand-written digit presents ambiguities.

A second remarkable aspect is the fact that the algorithm

Table 1: Summary of performance for four champion networks evolved in independent runs on a subset of the MNIST training database (1000 examples for each digit class).

	,										
Training set - Classification error											
run no.	class 0	class 1	class 2	class 3	mean						
run 1	1.30%	1.80%	11.20%	9.40%	5.93%						
run 2	1.80%	3.10%	8.20%	6.10%	4.80%						
run 3	0.80%	2.10%	10.30%	8.80%	5.50%						
run 4	0.70%	2.60%	8.50%	8.00%	4.95%						
mean	1.15%	2.40%	9.55%	8.08%	5.29%						

Table 2: Summary of performance for the four champion networks assessed using a subset of the MNIST database different than the training set (1000 examples for each digit class).

Test set - Classification error										
run no.	class 0	class 1	class 2	class 3	mean					
run 1	0.80%	2.40%	11.60%	8.50%	5.83%					
run 2	1.20%	4.10%	9.40%	6.40%	5.28%					
run 3	1.10%	2.30%	10.90%	8.50%	5.70%					
run 4	0.60%	3.60%	9.10%	6.40%	4.93%					
mean	0.93%	3.10%	10.25%	7.45%	5.43%					

evolved with a limited number of examples, 1000 for each digit class to be exact. The MNIST set, however, contains approximately 6000 examples for each digit. The error rates on the test set, composed of examples (1000 for each digit) that were not seen during evolution, indicate a clear generalization capability. Whereas most evolutionary algorithms tend to exploit the evolutionary environments and to perform less well with new inputs, the proposed POET method appears to generalize well. In particular, we observe that the drop in performance on the test set is extremely small. Tables 1 and 2 indicate that the network 1 has even slightly improved performance, whereas networks 2, 3 and 4 have very marginal drops in performance, 0.48%, 0.20%, and 0.24% respectively.

A possible explanation for this fact is that the gene expression mechanisms, from which weights are derived, perform distributed weight changes, rather than specific single-weight mutations. Therefore, rather than fitting single pixels in the input images (which may lead to overfitting), classifications may relay on larger pixel areas. Another hypothesis is that the size of the network (relatively small for such image classification tasks) contributed to reduce the gap in performance between the training and the test set. To sum up, more analysis is needed at this point to explain the high generalization capabilities of the proposed algorithm. Nonetheless, the evidence presented here suggests that the proposed algorithm may implicitly bring high generalization capabilities.



Figure 9: Image of the POET grid at the end of development, for one of the champions in the image classification evolutionary experiments. The grayscale-code indicates the contribution of the cell to weights. Elliptical areas outline the locations of several cell proliferations.

Discussion

The results described here demonstrate, for the first time in this particular domain, the full potential of indirect encoding in the optimization of large search spaces. In particular, the method shows that evolutionary search may leave the domain of toy problems and be applied to real world problems such as image recognition and classification. The rationale for the method performance is the ability of ET to devoevolve target shapes of any kind and size. In this work, we have replaced a geometric target with a computational one, and proved that the method still works. Nonetheless, more investigations with different network topologies is needed to assess the intrinsic mechanisms and rules of POET's search dynamics. The generalization capabilities, suggested by the similar error rates both in the training and in the test set, indicate that the developmental method may extract general features of the input spaces. This encouraging result calls for more analysis on the network topologies that emerge autonomously from the interaction of evolution and develop-

The absolute error rates on the training set (around 5%) are similar to error rates of other 2-layer neural networks with a similar number of hidden units (300) (LeCun and Cortes, 1998), although those rates refer to the complete MNIST dataset. It is important to observe that this study proposes a proof-of-concept on the use of evolutionary de-

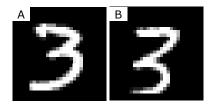


Figure 10: Examples of correct and incorrect classifications. (A) This digit was classified as 3 with the highest confidence. (B) This digit (labelled as 3 in the MNIST set) was classified as 2 with low confidence, the second guess was 3.

velopmental algorithms on image classification problems. A more optimized evolutionary process, for example involving larger populations or more generations (see Appendix), might lead to better performance. A second consideration is that the evolutionary developmental network might be used effectively as a starting point for further training with a standard approach (e.g., error backpropagation or Hebbian learning), possibly resulting in an improvement of the classification rate.

The process of development, directed by genetic instructions in stem cells, implies that the phenotype grows during a lifespan before reaching a mature state when it is capable to classify the input images. During development, the morphogenetic process in the current algorithm does not involve neural plasticity. Therefore, an exciting future direction consists in the combination of evolution, development, and plasticity. Particularly in the domain of image classification, in which learning algorithms for deep structures performed the best, the combination of development and learning is of high interest to understand general principles of learning in living organisms.

In this work the mapping of the 2D grid points to parameters (the weights) is initially done randomly and then allowed to change by means of swap events under genetic control. This mapping method is only one among many possible, and we are currently entertaining other possibilities. As the model is further improved, we plan to keep using swap events, because they help bring correlated parameters close to each other. Thanks to swap events such parameters can be optimized together. Otherwise they would have to be optimized independently, and encoded (in the case of the genetic algorithm) with separate genes.

A final consideration regards the possibile use of the optimization method proposed in this initial work. Here, POET was employed to optimize weights in artificial neural networks. Nevertheless, the method is not limited to this domain and, in fact, any number of parameters in a large search space can be optimized with the proposed method. This will be the matter of future work.

Conclusions

A new generative-developmental system for parameter optimization, named POET, is presented. The method is tested on the optimization of weights of a large neural network, evolved to classify handwritten digits from the MNIST database. The results show competitive error rates for evolutionary algorithms (although not for state-of-the-art learning algorithms), but more interestingly, they show impressive generalization capabilities on the test set, a remarkable result for evolutionary methods. To the best of our knowledge, the proposed method is the first purely evolutionary algorithm to search the neural network weights to classify the MNIST database. While this first test was performed on a subset of the MNIST database, and on a shallow neural network, the results are encouraging for the possible future application of POET to deep neural networks and more challenging classification problems.

Appendix

This section presents additional implementation details. We provide the source code of our software at www.evosys.org/software. Here a brief description of the key parameters of the algorithm.

For the experiments we used a grid size of 1000×1000 cells, in which an initial number of 2500 stem cells was injected. The maximum distance between stem cells was 10 grid points (a higher distance elicited the formation of new stem cells). Development was allowed to proceed for 50 developmental stages. In each stage a maximum number of 20 events (proliferation, apoptosis and swap combined) was allowed to take place. The maximum size of the event area was 200×200 cells. Each developmental stage was evolved in 1000 generations. Each neural weight was mapped to 3 different positions on the grid.

For the genetic algorithm, we used a population size of 512 distinct individuals. Each individual was composed of 1000 genes, and each gene was encoded with 500 base-4 digits, for a total genome size of 500000 digits. We used wheel selection with elitism (the best 64 genomes were copied without mutation in the next generation). We employed the genetic operators of mutation (with probability 0.005 per digit), and crossover (with probability 0.5 for each offspring genome).

Acknowledgements

This work was supported by the Polish National Science Center (project BIOMERGE, 2011/03/B/ST6/00399). This work was also partially supported by the European Communitys Seventh Framework Programme FP7/2007-2013, Challenge 2 Cognitive Systems, Interaction, Robotics under grant agreement No 248311 - AMARSi.

References

- Bengio, Y. (2009). Learning deep architectures for AI. *Foundations* and *Trends in Machine Learning*, 2(1):1–127.
- Bianconi, E., Piovesan, A., Facchin, F., Beraudi, A., Casadei, R., Frabetti, F., Vitale, L., Pelleri, M., Tassani, S., Piva, F., Perez-Amodio, S., Strippoli, P., and Canaider, S. (2013). An estimation of the number of cells in the human body. *Annals of Human Biology*, 40(6)2:463–467.
- Bongard, J. and Pfeifer, R. (2003). Evolving complete agents using artificial ontogeny. In *Morpho-functional Machines: The New Species*, pages 237–258. Springer.
- Cheney, N., MacCurdy, R., Clune, J., and Lipson, H. (2013). Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 167–174.
- Clune, J., Ofria, C., and Pennock, R. (2009). The sensitivity of HyperNEAT to different geometric representations of a problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 675–682.
- Cruz, M., Siden, A., Calaf, G., Delwar, Z., and Yakisich, J. (2012). The stemness phenotype model. *ISRN Oncology*.
- Cussat-Blanc, S. (2008). From single cell to simple creature morphology and metabolism. In *Proceedings of the 11th International Conference on the Simulation and Synthesis of Living Systems (ALife XI)*, pages 134–141.
- Eggenberger-Hotz, P. (1997). Evolving morphologies of simulated 3d organisms based on differential gene expression. In *Proceedings of the 4th International Conference on the Simulation and Synthesis of Living Systems (ALife IV)*, pages 205–213.
- Federici, D. (2004). Using embryonic stages to increase the evolvability of development. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) Workshop Program.*
- Fontana, A. (2008). Epigenetic tracking, a method to generate arbitrary shapes by using evo-devo techniques. In *Proceedings* of the 8th International Conference on Epigenetic Robotics (EPIROB).
- Fontana, A. (2009). Epigenetic tracking: biological implications. In *Proceedings of the 10th European Conference on Artificial Life (ECAL)*, volume 5777 of LNCS, pages 10–17.
- Fontana, A. (2012). Epigenetic Tracking: an evolutionarydevelopmental approach to generate 3-dimensional simulated structures. PhD thesis, Technical University of Gdansk, Gdansk, Poland.
- Fontana, A. and Wróbel, B. (2012). A model of evolution of development based on germline penetration of new "no-junk" DNA. *Genes*, 3:492–504.
- Fontana, A. and Wróbel, B. (2013a). An artificial lizard regrows its tail (and more): regeneration of 3-dimensional structures with hundreds of thousands of artificial cells. In *Proceedings of the 12th European Conference on Artificial Life (ECAL)*, pages 144–150.

- Fontana, A. and Wróbel, B. (2013b). Embryogenesis, morphogens and cancer stem cells: putting the puzzle together. *Medical Hypotheses*, 81(4):643–649.
- Gruau, F., Whitley, D., and Pyeatt, L. (1996). A comparison between cellular encoding and direct encoding for genetic neural networks. In *Genetic Programming 1996: Proceedings of the 1st Annual Conference*, pages 81–89.
- Hornby, G. (2005). Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pages 1729–1736.
- Joachimczak, M. and Wróbel, B. (2009). Evolution of the morphology and patterning of artificial embryos: scaling the tricolour problem to the third dimension. In *Proceedings of the 10th European Conference on Artificial Life (ECAL)*, volume 5777 of LNCS, pages 33–41.
- Koutník, J., Cuccu, G., Schmidhuber, J., and Gomez, F. (2013). Evolving large-scale neural networks for vision-based reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 1061–1068.
- Kumar, S. and Bentley, P. (2003). On growth, form and computers. Academic Press.
- LeCun, Y. and Cortes, C. (1998). MNIST handwritten digit database. AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist.
- Lindenmayer, A. (1968). Mathematical models for cellular interaction in development. *Journal of Theoretical Biology*, 18:280–315.
- Roesch, A., Fukunaga-Kalabis, M., Schmidt, E., Zabierowski, S., Brafford, P., Vultur, A., Basu, D., Gimotty, P., Vogt, T., and Herlyn, M. (2010). A temporarily distinct subpopulation of slow-cycling melanoma cells is required for continuous tumor growth. *Cell*, 141(4):583–594.
- Roggen, D. and Federici, D. (2004). Multi-cellular development: is there scalability and robustness to gain? In *Parallel Problem Solving from Nature-PPSN VIII*, pages 391–400. Springer.
- Smith, L. and Thelen, E. (1993). A dynamic systems approach to development: applications. The MIT Press.
- Stanley, K., Clune, J., D'Ambrosio, D., Green, C., Lehman, J., Morse, G., Pugh, J., Risi, S., and Szerlip, P. (2013). CPPNs effectively encode fracture: a response to critical factors in the performance of HyperNEAT. *University of Central Florida Dept. of EECS Technical Report CS-TR-13-05*.
- Stanley, K., D'Ambrosio, D., and Gauci, J. (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212.
- Stanley, K. and Miikkulainen, R. (2003). A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130.
- van den Berg, T. and Whiteson, S. (2013). Critical factors in the performance of HyperNEAT. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 759–766.
- Wolpert, L. and Ticke, C. (2010). *Principles of development*. Oxford University Press.

Abstract of: "On the Relationships between Generative Encodings, Regularity, and Learning Abilities when Evolving Plastic Artificial Neural Networks"

Jean-Baptiste Mouret and Paul Tonelli

Sorbonne Universités, UPMC Univ Paris 06, UMR 722, ISIR, F-75005, Paris, France CNRS, UMR 7222, ISIR, F-75005, Paris, France mouret@isir.upmc.fr

This paper is an abstract of (Tonelli and Mouret, 2013).

A major goal of bio-inspired artificial intelligence is to design artificial neural networks with abilities that resemble those of animal nervous systems. It is commonly believed that two keys for evolving nature-like artificial neural networks are (1) the developmental process that links genes to nervous systems, which enables the evolution of large, regular neural networks, and (2) synaptic plasticity, which allows neural networks to change during their lifetime. So far, these two topics have been mainly studied separately. The present paper shows that they are actually deeply connected.

One of the main challenge when designing Artificial Neural Networks (ANN) with learning abilities is to make them capable of learning in a large class of situations. When artificial evolution is used to design a plastic ANN, the topology of the networks is the result of the interactions between the fitness function, the encoding and the associated variation operators; but the topology critically constrains what can be learned.

The most classic approach to foster general learning abilities is to design a fitness function that tests each neural network in several test cases, and rewards individuals that successfully adapt their behavior to each of them. To ensure that networks possess general learning abilities, it is then required to assess their abilities to learn in a new set of test cases, that is, test cases that have never been encountered by the evolutionary process. The success of this "episodic fitness" approach relies on the assumption that if enough test cases are used, then it should become easier for the evolutionary process to design a generic structure than a specialized one.

Unfortunately, even in simplistic and constrained toy problems, many test cases need to be included in the fitness function to obtain general learning abilities. This approach is unlikely to scale-up to life-like neural-networks.

This is where developmental systems have a role to play. These systems evolve short descriptions of large structures by exploiting regularities observed in nature, such as repetition of useful sub-parts, symmetries, and symmetries with variation (Stanley et al., 2009; Mouret et al., 2010). They

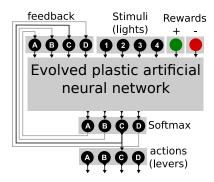


Figure 1: Formalization of the Skinner box as a task for a plastic artificial neural network.

more easily describe regular structures than irregular ones, because the former can be described by a few general rules whereas the latter require describing either each element, or a list of exceptions to general rules. As a consequence, developmental systems bias the search space towards regular structures (Clune et al., 2011). We here propose that this bias towards regularity is critical to evolve plastic neural networks that can learn in a large variety of situations. Intuitively, this bias makes it more likely to obtain generic networks that apply the same learning rules to whole sets of inputs, instead of networks that are finely-tuned to only solve the test cases used in the fitness function. A direct consequence is that using developmental systems to evolve plastic neural networks should facilitate the evolution of plastic ANNs with general learning abilities.

This hypothesis is tested using a simulated "Skinner box" (Figure 1), a classic experimental setup for operant conditioning in which a caged animal must learn to associate stimuli (e.g. lights) to actions (e.g. pushing a lever). If the animal executes the correct action, it is rewarded (e.g. by some food); if it chooses the wrong one, it is punished (e.g. by an electric shock). There is no delay in the reward, so there is no credit assignement problem.

The topology and the parameters of evolved ANNs are encoded with three different encodings, with three different levels of expected regularity. The first encoding, called

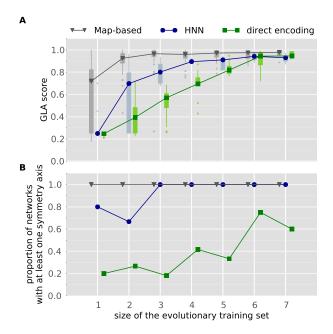


Figure 2: Relationship between encoding choice, general learning abilities, and size of the evolutionary training set (i.e. the set of test cases using in the fitness function).

A. Generative encodings yield plastic ANNs with better general learning abilities (GLA) than those evolved with a direct encoding. Morever, increasing the size of the evolutionary training set increases the general learning abilities. Each box extends from the lower to upper quartile values of the data, with a symbol at the median. B. Generative encodings yield more regular networks than a direct encoding, and increasing the size of the evolutionary training set increases the regularity of evolved networks.

the map-based encoding (Mouret et al., 2010), is inspired by computational neuroscience models in which ANNs are described as graph of single neurons and neural maps (spatially-organized identical neurons) that are connected with a few possible connection schemes (usually only oneto-one and one-to-all). This encoding produces very regular neural networks because it has to treat each neuron in a map in the exact same way as the other neurons of the same map. The second encoding is a simplified version of Hyper-NEAT (Stanley et al., 2009), called HNN, for Hyper Neural Network. The last encoding is a classic direct encoding in which evolution directly acts on the structure and the parameters of the ANN. We use neuro-modulated Hebbian plasticity (Soltoggio et al., 2008) and we estimate the regularity of networks by counting the number of automorphisms, that is, counting symmetry axes (Junttila and Kaski, 2007).

The results show a clear difference in the abilities of network obtained with each encoding to learn associations that were not tested in the fitness function (General Learning Abilities score: GLA score). With a direct encoding, the

GLA score grows linearly with the size of the evolutionary training set (Figure 2). The HNN encoding appears as a trade-off between the direct encoding and the map-based encoding: for each size of the evolutionary training set, the GLA score obtained with HNN is consistently higher than the one obtained with the direct encoding, yet it is lower than the one reached with the map-based encoding.

The experiments reported in this paper add weight to the hypothesis that using a developmental encoding improves the learning abilities of evolved, plastic neural networks. Complementary experiments reveal that this result is the consequence of the bias of developmental encodings towards regular structures (Clune et al., 2011): (1) encodings that tend to produce more regular networks yielded networks with better general learning abilities; (2) in our experimental setup, whatever the encoding is, networks that are the more regular are statistically those that have the best learning abilities. Overall, our experiments show that current generative encodings and neuro-modulated Hebbian plasticity make a promising combination to evolve large, plastic neural networks.

Acknowledgements

This work has been funded by the ANR Creadapt project (ANR-12-JS03-0009).

References

- Clune, J., Stanley, K. O., Pennock, R. T., and Ofria, C. (2011). On the performance of indirect encoding across the continuum of regularity. *IEEE Transactions on Evolutionary Computation*, 15(3):346–367.
- Junttila, T. and Kaski, P. (2007). Engineering an efficient canonical labeling tool for large and sparse graphs. In *Proc. of the 9th Workshop on Algorithm Engineering and Experiments and the 4th Workshop on Analytic Algorithms and Combinatorics*, pages 135–149.
- Mouret, J.-B., Doncieux, S., and Girard, B. (2010). Importing the computational neuroscience toolbox into neuro-evolution-application to basal ganglia. In *Proc. of GECCO*, pages 587–594. ACM.
- Soltoggio, A., Bullinaria, J. J. A., Mattiussi, C., Floreano, D., and Dürr, P. (2008). Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. In *Proc. of ALIFE*, pages 569–576.
- Stanley, K. O., D'Ambrosio, D. B., and Gauci, J. (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artificial life*, 15(2):185–212.
- Tonelli, P. and Mouret, J.-B. (2013). On the relationships between generative encodings, regularity, and learning abilities when evolving plastic artificial neural networks. *PLoS One*, 8(11):e79138.

Evolution, development and learning with predictor neural networks

Konstantin Lakhman and Mikhail Burtsev

Neuromorphic Cognitive Systems Lab, Kurchatov NBICS-Centre, National Research Center "Kurchatov Institute", 1 Kurchatov sqr., Moscow, Russia klakhman@gmail.com

Abstract

Study of mechanisms, that can make possible effective learning of artificial systems in complex environments, is one of the key issues in the adaptive systems research. In this paper we make an attempt to implement and test a number of ideas motivated by brain theory. Proposed model integrates evolutionary, developmental and learning phases. The main concept of this paper is the notion of predictor neural network which provide distributed evaluation of the effectiveness of goal-directed behavior on the neuronal level. We also propose learning mechanism based on gradually inclusion of new neuronal functional groups in case when the existing behavior fails to deliver adaptive result. We performed basic computational study of the model to investigate some of its' core properties such as evolution of innate and learned behavior and dynamics of the learning process.

Introduction

The problem of adaptation in multi-goal environments is a great challenge for state of the art machine learning. To solve it we may need to combine best practices from different frameworks as well as novel hypothesis from theoretical neuroscience.

There is a very little effort to integrate evolutionary and learning approaches to solve the problem of adaptive behavior in the field of machine learning despite the significant independent progress in both fields (e.g. neuroevolution (Stanley and Miikkulainen, 2002) and reinforcement learning (Sutton et al., 2011) frameworks). Roughly speaking most of the existing attempts to combine evolution and learning can be divided in two main categories: 1) evolution serves only for optimization of the initial controller, e.g. (Risi et al., 2010), or 2) evolution serves for generation of various kinds of "evaluation" functions, e.g. (Singh et al., 2009). In the latter case "evaluation" guides learning process during an agent's life by contributing to the prediction of the future states of the agent-environment system. From the standpoint of prediction mechanism it is possible to distinguish several approaches: prediction of the expected reward (Singh et al., 2009; Whiteson and Stone, 2006); explicit prediction of the expected parameters of an environment (Nolfi et al., 1994); generation of a separate "teacher"

structure (Nolfi and Parisi, 1996). Prediction in terms of the expected reward is not always possible in biological systems since in most cases there is no access to the reinforcement signals. Explicit prediction of an environment's parameters usually performed on the level of the whole controller. Therefore it is difficult to localize individual structural elements of the controller to be modified during learning.

At the same time within the field of theoretical neuroscience there are several suggestions for possible mechanisms underlying distributed formation of prediction in biological systems. However most of this approaches require carefully designed topology of neural microcircuits and give no clue how this topology could emerge in the evolution (Bastos et al., 2012; Hawkins et al., 2011).

Numerous researches in the artificial life field investigates interaction between evolution and learning (Ackley and Littman, 1991; Suzuki and Arita, 2003), including notable Baldwin effect. These works provide great insight into the theoretical understanding of the problem, however they usually utilize synthetic learning models that are not capable to scale well to the real-world problems.

What is required for the neural network learning algorithm to adapt successfully in a multi-goal environment? Starting from the theories of functional systems (Anokhin, 1974) and neuronal group selection (Edelman, 1993) we suggest the following logic to derive such requirements. The neural network should be able to produce initial or *primary* repertoire of basic behaviors. When during lifetime the agent starts to encounter problems in achieving goals with the primary behaviors then this initial repertoire should be extended to allow mission completion for a spectrum of environmental variations. These particular solutions acquired by learning constitute the secondary repertoire of behaviors. Primary behaviors can be generated by evolutionary and developmental algorithms or pre-specified by hand. During life-time adaptation learning algorithm should detect failures to execute existing behaviors and generate actions to create alternative solution. We suggest that failure detection should be distributed over the whole neural network. Our hypothesis is that such detection is possible when neurons not only activate but also predict future activity of each other. If some neurons detect mismatch between predicted and actual activities then the learning starts. We require that creation of a new action sequence should not disrupt existing behavior. This can be achieved by integration of new neurons in the network during learning process without changing already existing part of the network. Compared to the previous attempts to formalize principles of the functional systems theory (Red'ko et al., 2004; Komarov et al., 2010) our current model implements adaptive learning by distributed prediction on the level of individual neurons, so we call this architecture predictor neural network. In this paper we present computational study of the model consisting of evolutionary and developmental phases for generation of primary behavioral repertoire as well as life-time phase with learning controlled by continuous distributed prediction on the neuronal level.

The Model

Overview

According to the neuronal group selection theory (Edelman, 1993) developmental algorithm in our model targets two different objectives: 1) generation of a primary repertoire of behaviors and 2) generation of "diversity of repertoires" required for further self-learning during the life. During development an agent's genotype translates into the neural network controlling the agent's behavior in the environment. Diversity required for the selection of neuronal groups is implemented by simulation of the brain's anatomical regions with *neuronal pools*. Neurons in the same pool have similar but not identical connections with the rest of the network. During development endogenous stochastic activations of neurons lead to selection of highly connected subnetworks. These subnetworks produce primary behaviors of the agent. Remaining neurons form a set of silent neurons that are used for learning.

In the theory of functional systems (Anokhin, 1974) every goal-directed behavior serving some adaptive function unfolds in three stages: 1) generation of an action and prediction of its' results; 2) evaluation of results after action completion; 3) formation of a new functional system of neurons if additional leaning is needed. To implement both generation of action and evaluation of results neurons in our model have two types of synapses namely effector and predictor. Effector synapse is the same as in the standard artificial neural network. Predictor synapse does not propagate excitation but contributes to prediction of future activity of the neuron. Moments of the mismatch between expected results and state of the environment are detected in the neurocontroller as neurons calculate discrepancy between predicted and observed activity of each other. Detected mismatch initiates activation of silent neurons to modify agent's behavior. Agent's ability to generate meaningful and useful predictions at the neuronal level is subject to evolutionary selection of predictor connections in the network.

Multi-goal Environment and Neural Net

In the model population of agents evolves on the hypercube. State of the environment is represented by a binary string and at each discrete time step the agent can change the state of one bit. A goal in this environment is defined as the consecutive changes of a particular bits of the state vector. Set of such a goals in the environment forms a branched hierarchy. One goal could be nested in another one, i.e. be a beginning of it, and different goals could have identical starting nested goal. Detailed description of the environment can be found in (Lakhman and Burtsey, 2013).

The agent operates in the environment for a fixed amount of time. Fixed reward is associated with each goal and a value of reward accumulated over life-time affects the reproductive success of an agent at the evolutionary phase of the simulation. However, in contrast to the paper (Lakhman and Burtsev, 2013) we have changed how reward is recovered in the case when the agent attempts to reach the same goal during its' recovery period. Previously, the reward was reseted to zero on the goal's completion and then linearly recovered to the initial value. The agent had no information about accumulated reward. As we introduce learning in the current study there should be some feedback from the environment to the agent that completion of the goal failed. In the current version of the model completion of the goal during value recovery is impossible. If the agent tries to perform an action that would lead to the finalizing of the unrestored goal then the change of the bit is blocked and the state vector of the environment remained the same. Thus, the agent can perceive failure to complete target actions sequence by the neurons predicting the environmental change after successful action. Note that in the current scheme of agent-environment interaction the agent also has no direct information about the value of reward.

The structure of the agent's neurocontroller is determined by a processes of evolution, development and learning. The controller consists of three sets of neurons: input, output and interneurons. Number of input and output neurons is fixed and determined by a dimension $n^{\rm env}$ of the hypercube ($n^{\rm env}=8$ for all simulations). Input neurons represent current state vector with one neuron for each bit. Output neurons encode action that agent performs on a current time step. Each pair of output neurons is responsible for turning on or off a particular bit of the state vector. Current agent's action are selected according to the pair of the most active output neurons.

Interneurons of a neurocontroller are organized in layers with no restriction on connectivity. This allows to form both direct and recurrent effector synapses. Recurrent effector synapses propagate excitation with a unit time delay.

Evolutionary Algorithm

Genotype of an agent is defined by a tuple: G=(P,EC,PC), where $P=\{P_{\alpha}\}$ is a set of neuronal pools, $EC=\{ec_{\gamma\alpha}\}$ is a set of effector connections between pools, $PC=\{pc_{\gamma\alpha}\}$ is a set of predictor connections. During development this genotype is translated to the neural network. Each neuronal pool P_{α} of the genotype corresponds to a group of neurons with similar connectivity topology and has the following parameters: pool's size c_{α} , i.e. the number of neurons that will be formed from this pool during development; mean m_{α} and standard deviation σ_{α} of the biases of the neurons, that corresponds to a particular pool.

Each effector connection $ec_{\gamma\alpha}$ between pools α and γ has the following parameters: mean $m_{\gamma\alpha}$ and standard deviation $\sigma_{\gamma\alpha}$ of the weights of the synapses, that will be formed between neurons belonging to the corresponding pools; probability of the synapse development $p^{\rm dev}\left(ec_{\gamma\alpha}\right)$.

The only parameter for predictor connections $pc_{\gamma\alpha}$ is probability of connection development $p^{\rm dev}$ $(pc_{\gamma\alpha})$.

Reproductive success of the agent, i.e. the probability to be selected as a parent for the next population, is directly proportional to the value of reward accumulated over fixed amount of time. An offspring inherits parent's genotype transformed by mutations, including all numerical parameters of the genome introduced above. The structure of the network is modified by *duplication pool* mutation described in detail in the paper (Lakhman and Burtsev, 2013). This mutation substitute a single "parent" pool with two "offspring" pools with the same connectivity structure and half the size of the ancestor. We also used addition and deletion of effector and predictor connections as additional structural mutations. Full list of values of parameters that have been used for the simulations is provided in the Appendix.

Developmental Algorithm

Generation of neuronal pools Development starts with translation of the agent's genotype G = (P, EC, PC) into complete network. Each neuronal pool P_{α} is filled with c_{α} neurons. Bias b_i for the neuron v_i is drawn from a normal distribution: $b_i \sim \mathcal{N}\left(m_{\alpha}, \sigma_{\alpha}^2\right)$, $\forall v_i \in P_{\alpha}$. Effector synapse is created between neurons $v_i \in P_{\alpha}$ and $v_j \in P_{\beta}$ according to the pool connectivity coded in the genome: $w_{ji} \sim \mathcal{N}\left(m_{\beta\alpha}, \sigma_{\beta\alpha}^2\right)$ with probability $p^{\text{dev}}\left(ec_{\beta\alpha}\right)$, if $ec_{\beta\alpha} \in EC$; and $w_{ji} = 0$ otherwise (the synapse is not developing). As a result every neuron in the net has unique connectivity structure both in terms of topology and weights distribution. Development of predictor synapses occurs similarly with the only exception that predictor synapse has no weight.

Similar model of neuroevolution with pools encoded in the genome was proposed in the framework of *Enforced Sub-Populations* evolutionary algorithm (Gomez and Miikkulainen, 1999). However, according to this algorithm selection process was applied directly to neurons and only one neuron was selected from each pool in the development. Therefore this mechanism introduces competition between neurons with different specialization from the same pool. In the our model a similar approach is used to generate a number of variations of the same specialization within single pool.

Selection of primary neuronal groups The next phase of neurocontroller development has a goal to select neuronal groups for the primary behavioral repertoire. To perform this developmental selection we utilize competitive principle proposed in the theory of neuronal group selection (Edelman, 1993) which states that neuronal groups with highest endogenous stochastic activity are selected. As a result network of active neurons constitute neurocontroller at the beginning of agent's life. Less active neurons became silent but form required for further learning set of various local modifications of the network and can be recruited later.

Selection of primary neuronal groups takes place in the isolation from the model environment for a fixed number of time steps T^{sys} . At each time step each neuron of the network produces spontaneous activation with probability $p^{\rm spon}$. Outputs of the remaining neurons are calculated in the standard way using truncated positive sigmoid activation function (truncation implies zero output in case of the negative neuron's potential). Spontaneous activations of the neurons are designed to provide signal flow in the network in the absence of information about external environment. Total signed value of activation potential received by each neuron is calculated during simulation of endogenous network activity. Within each pool fraction p^{act} of the neurons is selected to participate in primary neuronal groups and remaining neurons become silent. The probability for the neuron to be selected is directly proportional to the corresponding value of accumulated potential.

The role of predictor synapses in the network is prediction of future activations of a neuron. If there is a predictor synapse between neurons v_i and v_j then pre-synaptic neuron's activity predicts that post-synaptic neuron will be active on the next time step (and vice versa in case of absence of activity). During development the predictor synapses between active neurons with prediction rate less then some threshold value $L^{\rm sig}$ (0.5 in the current study) are being deleted from the network. Predictor synapses connecting silent neurons remained intact.

It should be noted that the outputs of silent neurons during the agent's life are always set to zero, regardless of the value of potential they receive by incoming effector synapses.

Learning Algorithm

Innate behavior produced by initial neural network right after development is not optimal and should be complemented by learning. It is necessary to answer two basic questions when designing a learning algorithm: 1) When should learning begin? 2) How to perform learning if it is needed?

Mismatch detection To solve the first problem it is necessary to introduce mechanism for detection of the moments in which additional learning is needed. These are the moments when the agent performs actions that earlier led to adaptive result but now failed. Theory of functional systems (Anokhin, 1974) solves this problem by postulating that at the beginning of goal-directed behavior functional system of neurons (neural subnetwork responsible for some adaptive function) generates both the program of actions and prediction of expected outcomes so-called acceptor of action results (AAR). If AAR is not consistent with observed features of the environment, obtained immediately after the performance of the planned action, then functional system enters so-called mismatch state and initiates learning. Functional systems interact with an environment as well as with other functional systems. In our model predictor synapses make possible distributed evaluation of goal-directed behavior on the level of individual neurons. Several theoretical papers concluded that the biological neurons are able to provide similar functions (Fiorillo, 2008).

Let consider single neuron as a functional system. Signals from other neurons constitute its "environment". At a given time step the neuron can be excited or nonactive. In the model the neuron is in the excited state when its output is greater than zero. Each neuron forms prediction about its own future activity based on predictor synapses from other neurons. Each presynaptic neuron makes its contribution to the total prediction in accordance with the following scheme: if the presynaptic neuron was excited at time step t then it predicts that the postsynaptic neuron will be excited at time step t+1, and vice-versa if the presynaptic neuron was in the ground state. Thereby each neuron could calculate probability distribution of its own activity on the next time step based on predictor signals:

$$p^{\text{exc}}(v_{j}, t) = \frac{|\{v_{i} \mid ps_{ji} \in PS, o_{i}(t-1) > 0\}|}{|PS_{j}^{\text{act}}|}, \qquad (1)$$

$$p^{\text{sil}}(v_{i}, t) = 1 - p^{\text{sil}}(v_{i}, t)$$

where $p^{\mathrm{exc}}\left(v_{j},t\right)$ is the probability of neuron v_{j} of being active at time step $t,\ PS$ is a set of network's predictor synapses, o_{i} is the output of neuron $v_{i},\ PS_{j}^{\mathrm{act}}$ is a set of incoming predictor synapses of neuron v_{j} that are coming from active neurons. Neuron makes prediction based on this distribution only if one of these probabilities exceeds a fixed threshold $L^{\mathrm{pred}} \in (0.5,1]$. Eq. 1 implies that only active neurons affect prediction.

Described procedure assumes two types of the possible mismatch situations: I-type mismatch implies the absence of the neuron's activity when it was predicted and II-type mismatch implies the presence of the activity when it was not predicted.

Learning via specialization of "silent" neurons According to the systems-selection theory (Shvyrkov, 1986) learning at the neuronal level consists of neuronal specializations for the problem situation. This specialization occurs via selection of neurons from the "reserve" of low active cells or, in our case, from silent neurons.

Mismatch detection on the level of neurons allows effectively locate a specific place in a neural net where it is necessary to make modifications during learning. Thus for each neuron with I-type mismatch a silent neuron from the same pool with highest effector input is found and added to the set of network's active neurons. This just activated neuron is specialized on solving current problem. Effector synapses of this neuron are pruned to maximize recognition of the current neuronal input. This implies deletion of effector synapses from neurons that have not been excited on the current time step. Additionally we add a strong excitatory synapse $w \sim \mathcal{U}(0.5, 1)$ from activated neuron to the mismatched one. This synapse could potentially lead to the elimination of the mismatch on the postsynaptic neuron in the similar behavioral situations in the future. We also add additional predictor synapses from the neurons that predicted activation of the mismatched neuron to the set of predictor synapses of the activated neuron.

Learning for the II-type mismatch occurs in exactly the same manner except that we add strong inhibitory connection from activated to mismatched neuron in order to avoid mismatch in the similar behavioral situations.

As a result of learning the initial neural network is expanded by distributed integration of neurons specialized on solution of the current problem. This scheme fits into the conceptual framework considering learning as following the same principles as evolutionary process: generation of diversity and selection (Burtsev, 2008). The specialization of the silent neurons could be interpreted as a local mutations of neurocontroller, which can potentially lead to successful behavior. Nonetheless, learning terminates only if there is no mismatch between organism's expectations and actual state of environment detected at the neuronal level.

Experimental Results

As the first step we have studied how learning algorithm affects efficiency of an agents in terms of accumulated reward. 10 environments with different goals structure were generated randomly. Two modifications of the model – with or without learning (only with developmental phase) were run in every environment 5 times (50 runs in total for each version). Then we detected the best population in terms of the average accumulated reward in each run. For all agents in the best population development was performed 5 times with different random seeds. Finally, we run these 5 differ-

ent controllers from all $2^{n^{\rm env}}=2^8=256$ initial states of the environment. Resulting values of average reward are shown on the Fig. 1A. We have not found statistical difference between efficiency of the agents evolved with and without learning – t-test showed p-value = 0.23. However, if learning is switched off for the agents evolved with it then average reward decreases significantly indicating importance of learning in this case. Effect of learning is detailed on the Fig. 1B where majority of runs show lower reward without learning. As one can see several runs do not suffer from learnings shutdown, however this doesnt mean that learning didnt play significant role in these runs during the evolution.

We have also tested our learning mechanism against simple random activation of silent neurons (Fig. 1A). Random learning implies that silent neuron has a fixed probability of

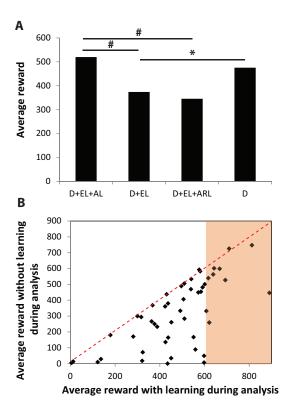


Figure 1: Effects of developmental and learning components of the model on reward. A) Average reward for the different types of agents (D – development, EL – learning during evolution, AL – learning during an analysis phase, ARL – "random learning" during an analysis phase). # denotes paired samples t-test with p–value < 0.001, * denotes t-test with p–value = 0.012. B) Comparison of the efficiency of the agents that evolved with learning and the same agents without learning. Each dot represents averaged value for the best population of corresponding evolutionary run. The range of reward values of the best 20 runs is highlighted.

integration into the network at a given time step. We randomized learning of the agents evolved with learning. Several activation probabilities were examined from the interval [0.01,0.1], but results are presented only for the best value of 0.5. Random specialization of neurons do not increase reward compared to normal learning and even has some tendency to decrease efficiency (paired samples t-test with p-value = 0.08) comparing to the results of the same agents but without any learning.

Results suggest that evolutionary algorithm alone is able to make non-learning agents almost as effective as learning ones. We found that after evolution the non-learning agents have in the neural net more pools and smaller pool sizes compared to the agents with learning (data is not shown). Larger pool sizes of learning agents indicate that selection support mechanism for variation of neuronal groups during learning.

As the next step we have studied evolution of the best run among those that use both developmental and learning phases in more details. For the best agent of each generation 5 different phenotypes were produced by randomly initialized development and then tested with and without learning. Resulting dynamics of innate and learned behaviors are presented on the Fig. 2A. Over the period of evolution an efficiency of learned behavior usually grows first and innate behavior follows. The increase of the best agent's reward after learning can occur both while efficiency of the innate behavior is decreasing (Fig. 2B) or remains unchanged (Fig. 2C).

As we have showed earlier (Lakhman and Burtsev, 2013) behavior evolved in the similar hypercubic environment consists of two phases: convergence phase when the sequence of actions depends on the particular initial state and stable repeating cycle of actions. We will call the latter behavioral cycle. Analysis of behavior cycles evolution for the best run shows that significant growth of learning efficiency is accompanied by explosion in variability (up to 200 variations) of behavioral cycles and subsequent change of the dominant behavioral strategy (data is not shown). The number of observed behaviors are sharply reduced when we analyze the

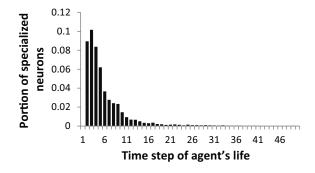


Figure 3: Dynamics of silent neurons specialization during an agent's life (averaged over the best 10 evolutionary runs).

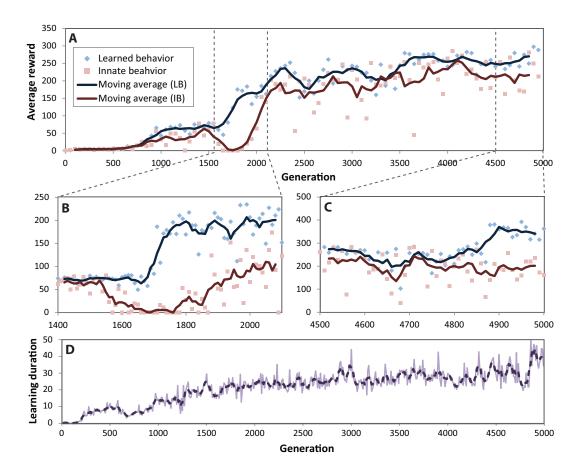


Figure 2: Evolutionary dynamics of innate and learned behaviors. **A)** Dynamics of the average rewards for the innate (pink) and learned (blue) behaviors for the best evolutionary run (agents were evaluated in the environment for 100 time steps and therefore rewards is roughly 2.5 times less than for the best run on the Fig. 1). The values are calculated for the best agent of the each 50-th generation. **B-C)** Periods when average reward after learning increases. The values are calculated for the best agent of the each 10-th generation. **D)** Averaged duration of learning. Duration of learning is calculated as the last time step of the agent's life when silent neurons specialize.

same agents but without learning. This might represent a general scenario when learning guide evolution of complex adaptive behavior in natural and artificial systems.

Reward accumulated by the agent during learning sometimes increases together with the average learning time, i.e. the period of silent neurons specialization(for example, starting from 4850-th generation on the Fig. 2D). Dynamics of the specialization of silent neurons during the learning process averaged over the best 10 evolutionary runs is shown on the Fig. 3. On the average about 50% of all silent neurons are turned on lifelong, but most of them are specialized at the beginning of life (up to 10-th time step). However, sometimes learning takes place even as late as 100-th time step for some of the evolutionary runs.

We found that there are the periods of evolution when success of learning significantly correlates with the number of specialized silent neurons (Fig. 4). This happens when behavior of the agents is not fully formed (i.e. the dominant

strategy is not stable) and learning can play significant role in improvement of partial actions sequences. Later in the course of evolution (for example in the 3500-th generation on the Fig. 2) this correlation disappears and roughly the same number of silent neurons are being activated regardless of the difference between learned and innate behavior (data is not shown). It seems that the role of learning is changing here, it mainly "corrects errors" that were made in the developmental process. On this stage the development can generate neurocontrollers that accumulate the same reward with or without learning.

To investigate how the neural network can generalize after learning we put the agent in each one of the states of the environment to learn then this agent with neural net after learning was tested starting from all the other states (Fig. 5A). For this analysis we have chosen the agent from the later stage of evolution (the best agent of 4950-th generation) and generated neurocontroller that shows big difference between re-

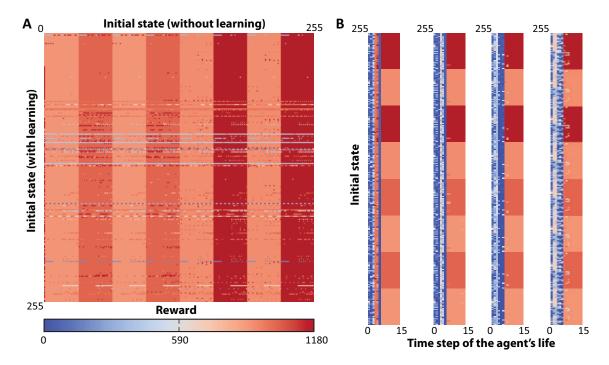


Figure 5: Neurocontroller performance after learning. **A)** After learning from each state of the environment the same learned neural network was started from each other state with the learning turned off. **B)** Performance of neurocontroller on the different stages of learning for a 4 random initial states(see text for the details).

All results presented on the figure are for the best agent of the 4950-th generation from the Fig. 2 (see text for more details).

wards with and without learning. Fig. 5A demonstrates that the neurocontroller after learning from one state of the environment is also efficient in the most cases when the agent starts life from the different state. "Stripes" on the figure correspond to the sets of initial states with different accumulated reward for the same neural net. This means that the agent achieves the same goals in the same order, but can

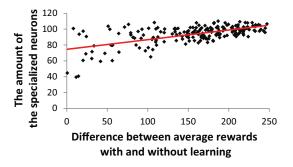


Figure 4: The number of specialized silent neurons positively correlates with the quality of learning. Each point represents value of reward averaged over all initial states of the environment for one development of the best agent of 1781-th generation from the Fig. 2. Spearman correlation coefficient $\rho=0.52$ with $p{\rm -value}<0.001$

be more or less optimal in terms of the number of actions performed.

The same agent that was used to produce Fig. 5A was utilized to study dynamics of the learning process (Fig. 5B). The agent was started with learning from a few states and then after each time step we evaluated efficiency of the controller, obtained so far, by running its non-learning version from all states of the environment. The results demonstrate that learning occurs only at the earliest stage of agent's life during the first 6–10 time steps. This is also consistent with the average dynamics of the specialization of silent neurons presented on the Fig. 3. Important feature of the learning process is that it is non gradual. Learning can even lead to temporary degradation of the intermediate controller performance. However when learning is finished the agent behavior abruptly becomes adaptive. It is important to notice that behavioral policies produced by intermediate controllers can be found in the behavior of the agents from earlier generations.

Conclusions

In this paper we presented a novel model of adaptive behavior that combines evolutionary, developmental and learning phases. In the model we introduced for the first time a number of principles inspired by brain theory: 1) selection of neuronal groups during development to build primary net-

work structure; 2) evolution of neuronal anatomy instead of exact neuronal connectivity to create diversity of subnetworks for learning during life-time; 3) distributed prediction mechanism that makes possible to detect mismatch between expected and perceived states of the environment for the initiation of learning on the neuronal level; 4) learning by specialization of "silent" neurons that produces non-disruptive modification of the network structure suitable for the acquisition of alternative behaviors in multi-goal environments.

In computational study we attempted to reveal some basic properties of this model, such as dynamics of the innate and learned behavior in evolution, dynamics of learning in terms of controller and behavior modification. Despite the fact that evolutionary algorithm "finds way" to make non-learning agents almost as effective as learning ones, our results demonstrate the important role of learning in evolution.

Acknowledgements

This work was partially supported by RFBR grant #13-04-01273.

References

- Ackley, D. H. and Littman, M. L. (1991). Interactions between learning and evolution. volume 10, pages 487–509.
- Anokhin, P. (1974). Biology and Neurophysiology of the Conditioned Reflex and Its Role in Adaptive Behavior. Pergamon, Oxford.
- Bastos, A., Usrey, M. W., Adams, R., Mangun, G., Fries, P., and Friston, K. (2012). Canonical microcircuits for predictive coding. *Neuron*, 76(4):695 – 711.
- Burtsev, M. (2008). Basic principles of adaptive learning through variation and selection. In Bullock, S., Noble, J., Watson, R., and Bedau, M. A., editors, *Proceeding of the ALIFE XI*, ALIFE XI, pages 88–93. MIT Press, Cambridge, MA.
- Edelman, G. M. (1993). Neural darwinism: Selection and reentrant signaling in higher brain function. *Neuron*, 10(2):115 125.
- Fiorillo, C. D. (2008). Towards a general theory of neural computation based on prediction by single neurons. *PLoS ONE*, 3(10):e3298.
- Gomez, F. J. and Miikkulainen, R. (1999). Solving non-markovian control tasks with neuroevolution. In *Proceedings of the IJ-CAI'99*, volume 2 of *IJCAI'99*, pages 1356–1361, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Hawkins, J., Ahmad, S., and Dubinsky, D. (2011). Htm cortical learning algorithms (white paper).
- Komarov, M., Osipov, G., and Burtsev, M. (2010). Adaptive functional systems: Learning with chaos. *Chaos*, 20(4):045119.
- Lakhman, K. and Burtsev, M. (2013). Neuroevolution results in emergence of short-term memory in multi-goal environment. In *Proceeding of the GECCO '13*, GECCO '13, pages 703–710, New York, NY, USA. ACM.
- Nolfi, S., Elman, J. L., and Parisi, D. (1994). Learning and evolution in neural networks. *Adaptive Behavior*, 3(1):5–28.

- Nolfi, S. and Parisi, D. (1996). Learning to adapt to changing environments in evolving neural networks. *Adaptive Behavior*, 5(1):75–98.
- Red'ko, V. G., Prokhorov, D. V., and Burtsev, M. S. (2004). Theory of functional systems, adaptive critics and neural networks. In *Proceeding of the IEEE International Joint Conference on Neural Networks*, volume 3, pages 1787–1792. IEEE.
- Risi, S., Hughes, C. E., and Stanley, K. O. (2010). Evolving plastic neural networks with novelty search. *Adaptive Behavior*, 18(6):470–491.
- Shvyrkov, V. B. (1986). Behavioral specialization of neurons and the system-selection hypothesis of learning. In *Human Memory and Cognitive Capabilities*, pages 599–611. Elsevier, Amsterdam.
- Singh, S., Lewis, R., and A.G., B. (2009). Where do rewards come from? In Taatgen, N. and van Rijn, H., editors, 31st Annual Conference of the Cognitive Science Society, pages 2601–2606, Austin, TX.
- Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10:99–127.
- Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. (2011). Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *Proceeding of the AAMAS '11*, volume 2, pages 761–768, Richland, SC.
- Suzuki, R. and Arita, T. (2003). The baldwin effect revisited: Three steps characterized by the quantitative evolution of phenotypic plasticity. In Advances in Artificial Life, volume 2801 of Lecture Notes in Computer Science, pages 395–404. Springer Berlin Heidelberg.
- Whiteson, S. and Stone, P. (2006). Evolutionary function approximation for reinforcement learning. *The Journal of Machine Learning Research*, 7:877–917.

Appendix

We used the following parameters of the artificial environment in all simulations: dimensionality of the hypercube environment n^{env} = 8 bits; agent lifetime duration T^{life} = 250 time steps; recovery time of a goal reproductive value $T^{\text{rec}} = 30$ time steps; probability of random change of the state vector's bit $p^{\text{stoch}} = 0.0085$ (for each bit); reproductive value, associated with every goal R = 20. Evolutionary algorithm was run with parameters: population $N^{\rm p}$ = 250 agents; period of evolution T^{ev} = 5000 generations; probability of the effector connection weight mutation $p^{\text{weight}} = 0.6$ (for each synapse); variance of the mutation of mean of effector connection weight $d^{\text{weight-mean}} = 0.08$; variance of the mutation of variance of effector connection weight $d^{\text{weight-variance}} = 0.08$; probability of adding an effector connection $p^{\rm add-con} = 0.1$ (for the whole network); probability of deleting an effector connection $p^{\rm del-con}$ = 0.05 (for the whole network); probability of adding an predictor connection $p^{\text{add-pred-con}} = 0.1$; probability of deleting an predictor connection $p^{\text{del-pred-con}} = 0.05$; probability of pool's duplication $p^{\text{dup}} = 0.007$ (for the whole network). Developmental algorithm was run with parameters: duration $T^{\text{sys}} = 200$ time steps; probability of neuron's spontaneous activation $p^{\text{spon}} = 0.1$; fraction of activated neurons $p^{\text{act}} = 0.4$ (for each pool); minimum prediction rate $L^{\text{sig}} = 0.5$. The mismatch threshold L^{pred} used in the learning algorithm was set to 0.5.

Online Extreme Evolutionary Learning Machines

Joshua E. Auerbach¹, Chrisantha Fernando² and Dario Floreano¹

¹Laboratory of Intelligent Systems, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland ²School of Electronic Engineering and Computer Science, Queen Mary, University of London, London, UK joshua.auerbach@epfl.ch

Abstract

Recently, the notion that the brain is fundamentally a prediction machine has gained traction within the cognitive science community. Consequently, the ability to learn accurate predictors from experience is crucial to creating intelligent robots. However, in order to make accurate predictions it is necessary to find appropriate data representations from which to learn. Finding such data representations or features is a fundamental challenge for machine learning. Often domain knowledge is employed to design useful features for specific problems, but learning representations in a domain independent manner is highly desirable. While many approaches for automatic feature extraction exist, they are often either computationally expensive or of marginal utility. On the other hand, methods such as Extreme Learning Machines (ELMs) have recently gained popularity as efficient and accurate model learners by employing large collections of fixed, random features. The computational efficiency of these approaches becomes particularly relevant when learning is done fully online, such as is the case for robots learning via their interactions with the world. Selectionist methods, which replace features offering low utility with random replacements, have been shown to produce efficient feature learning in one class of ELM. In this paper we demonstrate that a Darwinian neurodynamic approach of feature replication can improve performance beyond selection alone, and may offer a path towards effective learning of predictive models in robotic agents.

Introduction

The notion of the brain as fundamentally a prediction machine is an old idea (Helmholtz, 1860) that has recently been gaining traction within the cognitive science community (see e.g. Clark, 2013; Hawkins and Blakeslee, 2007). A consequence of this idea is that if we wish to build robots capable of exhibiting intelligent behavior, and adapting to different circumstances, then prediction must be a fundamental part of their cognitive architecture. If the world behaved in a fundamentally linear way, than this would be an easy problem to solve: linear models operating directly on raw sensorimotor data could be learned in a straightforward and efficient manner. Unfortunately, the world is noisy and full of non-linear interactions that make learning difficult (Enns,

2010). In order to make accurate, non-linear predictions it is necessary to find appropriate data representations from which to learn. Finding such data representations or features is a fundamental challenge for machine learning.

Often the domain knowledge of human experts is leveraged to design useful features for specific problems (LeCun et al., 1998). While this may be an effective means of making learning tractable in many instances, it is not an ideal solution. Using human expertise is problem-specific, expensive, injects potentially sub-optimal biases into the solution, and for many robotics applications (especially those employing soft materials or other unconventional components, e.g. Germann et al., 2014) the relevant expertise may not exist. For these reasons, learning representations in a domain independent manner is highly desirable.

One common method of predicting non-linear relationships is to train multilayer feed-forward neural networks, which have been proven to be universal function approximators (Cybenko, 1989; Hornik, 1991). Most frequently these networks are trained offline from a pre-compiled training-set of input and target output values by gradient-descent via the backpropagation algorithm (Rumelhart et al., 1988). This offers one approach to feature learning: by backpropagating the supervised error signal, features can be adjusted in the gradient-descent direction. While this method may work successfully in many applications, it often learns slowly and may require large datasets to be effective (Ciresan et al., 2010).

Many other approaches for automatic feature extraction have been proposed in the literature. One approach that has recently proven quite successful involves the use an unsupervised "pre-training" step followed by further refinement through error backpropagation (Hinton and Salakhutdinov, 2006). However, this method is computationally expensive—usually involving extended computation time even when specialized hardware is employed. Moreover, the necessity of doing extensive "pre-training" on a data set cannot be applied when learning must be done fully online.

In online learning, data is learned from as it is received. In this regime, previously seen data points cannot be revisited, and training gradients must be estimated from one (or a small subset) of the most recently seen data points. However, learning online from data as it is obtained is crucial in robotics domains where it is not possible to collect data *a priori* to be used in an offline batch mode. Moreover, even if possible, batch learning is not always desirable, because robotic agents may be operating in non-stationary environments within which they must continuously adapt. Finally, the volume of sensorimotor data obtained by agents may easily exceed the storage capacities of their onboard computers, especially if the agents continuously operate for extended time periods. For these reasons, this work concerns itself with online learning of predictors.

An alternative approach to the above methods, which has proven surprisingly effective, both for offline as well as online learning, is to randomly generate a large number of features on which to learn. Extreme Learning Machines (ELMs) (Huang et al., 2012) are a recently introduced formalization of single-hidden-layer feed-forward neural networks, where the feature-mappings are not tuned, but rather are chosen stochastically. This has the advantage that the only model parameters that are trained are the connection weights from hidden units to outputs, therefore simplifying learning to a linear regression problem¹. The intuition here is that these fixed random features create a dimensionality expansion on top of which it is often possible to fit a linear model.

Recent work has demonstrated that it is possible to automatically search for effective features in online learning scenarios through a generate and test procedure (Mahmood and Sutton, 2013). In that work, it was demonstrated that, in a form of ELM-like artificial neural network (ANN), predictive accuracy could be greatly improved by regularly discarding features that offer low utility and replacing them with new stochastically generated features. This is essentially a selectionist approach, whereby poor features are selected for elimination and new features are generated in a completely random manner devoid of any information transfer.

In this work we extend Mahmood and Sutton's approach by taking inspiration from the Neuronal Replicator Hypothesis (Fernando et al., 2010, 2012), which posits that "replication (with mutation) of patterns of neuronal activity can occur within the brain using known neurophysiological processes." Specifically, instead of introducing new features at random as Mahmood and Sutton have done, new features are created through a Darwinian process of replication + variation of existing features, which have been dis-

covered to be useful for solving the given prediction problem². We dub this learning architecture an Online Extreme Evolutionary Learning Machine (OEELM). We demonstrate that OEELMs are capable of achieving lower error than the purely selectionist approach employed in (Mahmood and Sutton, 2013) with a much smaller number of features. Moreover we demonstrate that this method compares favorably to backpropagation.

The remainder of this paper is structured as follows. The following section describes the OEELM method, and describes the experimental setup used for comparing this approach to existing methods. Next, the results of these experiments are presented and analyzed. A discussion of this method is then presented, followed by conclusions and directions for future research.

Methods

Taking inspiration from Mahmood and Sutton (2013), we investigate the problem of automatically searching for useful features in a fully online learning scenario. In this formulation, an ANN is attempting to learn by adjusting its parameters to better fit the observations emanating from a noisy data stream. The underlying learning architecture is an ELM-like, single-hidden-layer feed-forward ANN. Following the implementation described in (Mahmood and Sutton, 2013) the ANN architecture consists of an input layer fully connected to a hidden layer with nonlinear activation functions (all features have access to all inputs), which is then fully connected to a linear output unit that produces a prediction of a target value.

The nonlinearities in the hidden layer are achieved by means of Linear Threshold Units (LTUs) adopted from (Sutton and Whitehead, 1993). Specifically, the output of feature *i* is given as follows:

$$f_i(x^{(t)}) = \begin{cases} 1 & \text{if } \sum_{j=1}^m v_{ji}^{(t)} x_j^{(t)} > \theta_i^{(t)} \\ 0 & \text{otherwise} \end{cases} \forall i = 1, ..., n \quad (1)$$

for a network with m inputs and n features. Here, $x^{(t)}$ is the input vector at iteration t, $v_{ji}^{(t)}$ is the weight from input j to feature i at iteration t, $x_j^{(t)}$ is the jth component of $x^{(t)}$, and $\theta_i^{(t)}$ is the threshold of feature i at iteration t.

The prediction of the network at iteration t is given by

$$\hat{y}^{(t)} = \sum_{i=0}^{n} w_i^{(t)} f_i(x^{(t)})$$
 (2)

where $f_0(x)$ is a bias feature that is always set to 1.

At each iteration t, the network is presented with a single observation $(x^{(t)}, y^{(t)})$ from a noisy data stream and the

¹Echo State Networks (Becker and Obermayer, 2003) and Liquid State Machines (Maass et al., 2002) (collectively known as Reservoir Computing) employ a similar idea for recurrent neural networks: the output connections from a dynamical reservoir of stochastically generated neurons is trained to fit a teaching signal via linear regression

²It is worth stressing that, here, we evolve a population of features for a single predictor, rather than a population of predictors as in (Arthur, 1994; Bongard and Lipson, 2007).

output weights w are updated in order to reduce the mean squared error between the observed target value $y^{(t)}$ and the predicted value $\hat{y}^{(t)}$ by means of the Delta-rule (Widrow and Hoff, 1960):

$$\Delta w_i^{(t)} = \eta(y^{(t)} - \hat{y}^{(t)}) f_i(x^{(t)})$$

$$w_i^{(t+1)} = w_i^{(t)} + \Delta w_i^{(t)}$$
(4)

$$w_i^{(t+1)} = w_i^{(t)} + \Delta w_i^{(t)} \tag{4}$$

where η is a free parameter known as the learning rate.

Feature Selection

Mahmood and Sutton (2013) demonstrated that the predictive accuracy of this class of model could be improved if, instead of using a fixed random set of feature weights v, selection is employed to discard features offering low utility and replace them with new features during the course of online learning. One problem with implementing such a selectionist method in an online setting is that it may be difficult to quantify the utility of individual features. As argued in that work, if a batch learning system is optimized until convergence (see e.g. Schmidhuber et al., 2007) then the utility of features can easily be evaluated, but in an online setting this evaluation must be able to function on a per-example basis. Additionally, it is argued in Mahmood and Sutton (2013) that in online settings, where new data points are constantly arriving, the process of evaluating feature utilities must be computationally efficient and not add to the overall computational complexity of the learning system.

Mahmood and Sutton (2013) present a method that overcomes these limitations. For each iteration of online learning, a small fraction ρ of existing features is selected for elimination and replaced with newly generated, random features. They demonstrate that such an approach can be implemented such that the total computational complexity is no greater than a base system that implements the Deltarule for learning the readout weights: O(mn) for computing the output of the ANN, and O(n) for updating the readout weights. The reader is referred to that work for further details of this derivation.

Finally, that work presented three alternative approaches for estimating the utility of a feature in the online learning scenario. All three of the approaches are based upon the idea that the relative utility of a feature is related to the magnitude of its readout weight. The intuition behind this idea is that the magnitude of a feature's readout weight determines how much that feature contributes to the output of the network. Since the readout weights are trained to approximate the observed data, the magnitude of a feature's readout weight will therefore serve as a proxy for how much that feature serves to explain the observations.

The problem with using the actual readout weight magnitude of a feature is that newly introduced features will initially have small output weight magnitudes³, and without a

mechanism to allow them time to prove their usefulness they will immediately be selected for elimination. Here, we adopt one of the procedures described in that work for overcoming this difficulty. This is described next.

In the employed approach, the utility of a feature u_i is calculated as an exponential moving average (EMA) of the magnitude of its readout weight:

$$u_i^{(t+1)} = \alpha_u u_i^{(t)} + (1 - \alpha_u) |w_i^{(t+1)}|$$
 (5)

where α_u is the decay rate of the EMA, here chosen through a tuning procedure to be 0.9.

When a new feature k is introduced, u_k is set to the median utility value of all features so that it does not get replaced immediately. If this new feature is not useful then its actual readout weight w_k will remain near zero, and therefore u_k will shrink over time. This will lead to feature keventually getting replaced. On the other hand, if feature k is useful then w_k will increase in magnitude and the feature will remain in the network. We choose this particular procedure because it performed competitively with the other approaches described in (Mahmood and Sutton, 2013), and is straightforward to implement.

Feature Evolution

The technique of (Mahmood and Sutton, 2013) described above is purely selectionist: features with poor utility are selected for elimination and are replaced with features that are generated at random. However, it is known that purely selectionist search methods have limitations that may be overcome through the use of replicators (Fernando et al., 2010). Additionally, there is a growing body of evidence which suggests that there is a process of replication occuring within individual brains (Fernando et al., 2010, 2012). Taking these ideas as inspiration, we suggest that a Darwinian process of feature evolution (with replication) will be a more powerful search method than the purely selectionist approach.

Extending the above selectionist method into a Darwinian one is fairly straightforward. The process of estimating feature utility and selecting features for removal remains unchanged. The main difference is that this utility estimation becomes the fitness function on which an online (steadystate) evolutionary algorithm operates. Now, instead of introducing new features purely at random, eliminated features are replaced with mutated copies of other features, which have themselves proved to be useful for explaining the observed data.

Specifically, when a feature is selected for removal, a binary tournament is conducted to choose a "parent" feature for reproduction. Two features from the population are chosen at random and the one with higher fitness creates a copy of itself. Each gene of that feature (the weights v_{ii}) are then

³Newly introduced features are initially given a readout weight

of 0 so that they do not to contribute to the prediction before the learner has a chance to adjust this weight.

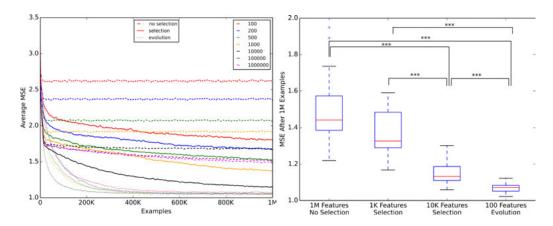


Figure 1: Comparison of learning errors across regimes and number of features. Left: this plot depicts how predictive-error (calculated as the mean across 30 independent runs of a sliding window estimate of mean squared error) varies over training time for all experimental setups investigated. Right: boxplots comparing the predictive-errors across regimes after being exposed to 1,000,000 examples. Asterisks denote statistical significance (*** = p-value < 0.001, Mann-Whitney U test with Bonferroni correction).

mutated with probability $p_{\rm mutate}$ and the resulting feature replaces the removed one. As is the case in the selectionist method, the readout weight of this feature is initialized to 0, and its fitness is initialized to the median fitness of the population. Conducting a binary tournament takes constant time (since the fitnesses are already in memory) and mutating the winner takes no more time than creating a new feature at random: O(m). Therefore, the evolutionary approach is no more computationally expensive than the selectionist one.

Experiments

We first reproduce the results presented in (Mahmood and Sutton, 2013) before comparing the selectionist approach with the evolutionary approach. These experiments are described here.

Refer back to Eqn. 1. In these experiments the system is assumed to take a binary input vector $x \in \{0,1\}^m$ and produce a scalar prediction $\hat{y} \in \mathbb{R}$. The input weights v_{ji} are initialized with either +1 or -1 uniformly at random. The threshold θ_i is set as $\theta_i = m\beta_i - S_i$, where S_i is the number of negative input weights of feature i and β_i is a free parameter. This formulation ensures feature i activates when at least $m\beta_i$ input bits match the prototype defined by the feature's weights. Following the procedure of (Mahmood and Sutton, 2013)⁴ β_i is set to $0.6 \ \forall i$.

In the absence of any feature search procedure the values of these parameters will remain fixed for the entire duration of an online learning task—the network is essentially an ELM with binary features. However, when either the selectionist or the evolutionary approach is employed, its task is to find a set of features that is appropriate for explaining the observed data.

The online learning task is conducted in simulation. At each iteration a binary, 20-dimensional input vector $x^{(t)}$ is generated uniformly at random. This input is fed through an ANN of the type described above containing 20 fixed random LTU target features f_i^* each having threshold parameter $\beta_i=0.6$. Next, the outputs of these features are linearly combined with output weights drawn from a normal distribution having mean 0 and unit variance $(w_i^* \sim N(0,1))$. The output of this network is then injected with Guassian noise $\epsilon_t \sim N(0,1)$ drawn independently at random for each iteration. Summarizing, the target value at iteration t, y_t is computed as:

$$y_t = \sum_{i=1}^{20} w_i^* f_i^* + \epsilon_t \tag{6}$$

The Gaussian noise makes the task more resemble real-world online learning tasks such as those found in robotics applications, and implies that if the learning network exactly learns the target function than the expected value of its mean squared error will be 1.

At each iteration, the learning rate η is set to $\frac{\gamma}{\lambda^{(t)}}$, where $\gamma \in (0,1)$ is the effective learning rate⁵, and $\lambda^{(t)}$ is an EMA estimate of the squared norm of the feature vector:

$$\lambda^{(t)} = \alpha_{\lambda} \lambda^{(t-1)} + (1 - \alpha_{\lambda}) (f^{(t-1)} \cdot f^{(t-1)}) \tag{7}$$

Unless otherwise specified, all reported results employ a decay rate $\alpha_{\lambda}=0.999$, and an effective learning rate $\gamma=0.1$.

Results

The above online learning problem is investigated under several experimental regimes. The first regime: "no selec-

⁴Clarified in (Mahmood, 2014).

⁵Called the *effective step-size* in Mahmood and Sutton's terminology.

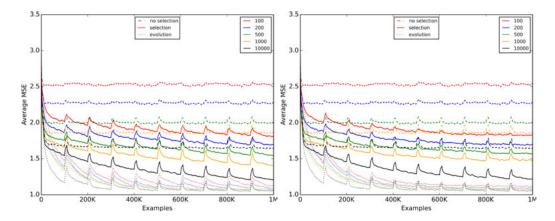


Figure 2: Exploring non-stationary environments. Here the environment (the target function) switches between two randomly generated functions every 100,000 iterations. Left: the same learning algorithms as are used for stationary environments. Right: for both the "selection" and "evolution" regimes features may be added to a growing archive that is immune from selection. These plots depict how predictive-error (calculated as the mean across 30 independent runs of a sliding window estimate of mean squared error) varies over training time.

tion" employs a fixed, random feature set, as is traditionally used in ELMs. The second regime: "selection" uses the purely selectionist approach of Mahmood and Sutton (2013). The third regime: "evolution" uses the evolutionary approach based on Darwinian neurodynamics introduced above (OEELMs). Each regime is investigated for different, fixed, number of features n. For each regime and value of n, 30 independent runs of the online learning scenario are conducted. Each learning scenario lasts for 1,000,000 iterations.

Under both the "selection" and "evolution" regimes, the fraction $\rho=0.005$ of the existing features having lowest estimated utility are selected for elimination at each iteration. Under the "evolution" regime, the input weights of copied features are each mutated with probability $p_{\rm mutate}=0.1$, chosen to optimize performance. Reported results are robust to small variations of this value.

Fig. 1 compares the performance of these regimes as the learning experiments progress. On the left, the accuracy of each experimental setup (regime and n value) is plotted for each iteration t as follows. Within each run, the current mean squared error (MSE) is estimated using a sliding window approach: the current error estimate is the mean of the individual squared errors of the past 10,000 data points. Due to the noise inherent in the data stream, the sliding window provides a better estimate of a predictor's accuracy at a given time than its error on any individual data point. The means of these MSE estimates are then taken across the 30 independent runs of each experimental setup. On the right, we show a boxplot comparing the most relevant final MSE estimates after 1,000,000 iterations.

With a fixed, random feature set (the "no selection" regime) performance improves as a function of the number of features, but continuing to increase the feature count

has diminishing returns. As demonstrated by Mahmood and Sutton (2013), and confirmed here, a purely selectionist approach to searching for features (the "selection" regime) with only 1,000 features can outperform a fixed feature set of 1,000,000 features. However, by using the OEELM approach described above, near optimal error can be achieved very rapidly. Moreover, using only 100 features with "evolution" not only outperforms all "no selection" formulations investigated, but also all "selection" formulations as well. Using 100 features with "evolution", the estimated MSE becomes significantly smaller (p-value < 0.001, Mann-Whitney U test) than that of all "no selection" and "selection" setups by iteration 104,900 and remains that way for the duration of the learning scenarios.

Non-stationary Environments

Often robotic agents are operating in non-stationary environments to which they must continuously adapt. In order to investigate whether feature evolution is also useful under changing environmental conditions the above online learning task is altered as follows. Instead of having a single target function that a network is attempting to predict, two different target functions are created. Since it is likely that different environmental conditions will have many similarities (e.g. the laws of physics remain unchanged across environments), the two functions are related to each other. Specifically, target function 1 is constructed exactly the same way as described above. Target function 2 is constructed from target function 1 by replacing 25% of its hidden nodes. For each node to be replaced, a new input weight vector is created uniformly at random, and a new readout weight is chosen from a Gaussian distribution with zero mean and unit variance.

The experimental procedure above is repeated for this

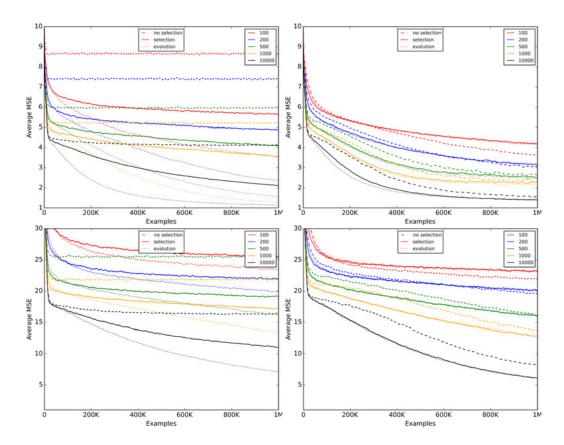


Figure 3: Increasing problem difficulty and combining feature evolution with backpropagation. Top: first, the difficulty of the problem is increased by going from 20 to 100 hidden units in the target function (left). Then the online learning algorithms are rerun in combination with a modified backpropagation procedure (right, see text for details). Bottom: The difficulty of the problem is increased further by using a target network with 500 hidden units, and the above is repeated. These plots depict how predictive-error (calculated as the mean across 30 independent runs of a sliding window estimate of mean squared error) varies over training time for this more difficult task.

new, non-stationary objective. First data points are sampled from target function 1 for the first 100,000 iterations, then data points are sampled from target function 2 for the next 100,000 iterations, and the target function continues to oscillate in this fashion for the duration of the task. As before, the data stream is noisy: target values are corrupted through the addition of Gaussian noise ($\epsilon_t \sim N(0,1)$).

Fig. 2 (left) plots how estimated MSE (again taken as the mean across the 30 independent runs per setup of the sliding window estimate) varies by regime and n value for non-stationary environmental conditions. Once again the "evolution" regime with only 100 learnable features outperforms both the "no selection" and "selection" regimes using many more features. Here, the ("evolution", 100) setup achieves significantly smaller error (p-value < 0.001, Mann-Whitney U test) than all "no selection" and "selection" setups after being trained on 176,100 examples and continues to have significantly smaller predictive error for the duration of the learning scenarios.

Performance of both the "selection" and "evolution"

regimes can be improved even further (especially for small numbers of learnable features) by incorporating a growing archive of useful features. This archive is initially empty. At each iteration there is a small probability ($p_{\rm archive} = \frac{n}{1,000,000}$) that the best, currently non-archived feature is added to the archive. Once a feature has been added to the archive it is no longer eligible for replacement. So, even if a feature is not currently useful it may stick around if it was found useful in the past—the archive is a growing, long term memory of canalized features. The performance for the non-stationary task with the archive included for both the "selection" and "evolution" regime is depicted in Fig. 2 (right) (the "no selection" results are the same as in Fig. 2, left).

Backpropagation

Mahmood and Sutton (2013) suggested that combining feature selection with updating feature weights in the gradient descent direction via backpropagation (Rumelhart et al., 1988) could achieve better performance than either method alone. Following their example, here we explore how feature

evolution might synergize with backpropagation. Specifically, we employ the modified version of backpropagation introduced in that work: the output weights are adjusted using the Delta-rule as above, but since LTUs do not vary continuously, the gradient of each hidden unit is estimated using a logistic function centered around the threshold of its LTU. Then, instead of using the magnitude of the error multiplied by the output weights $(y^{(t)} - \hat{y}^{(t)})w_i^{(t)}$ when computing the feature gradients, only the sign of this quantity is taken, so that feature weights v_{ji} are updated as follows:

$$\begin{array}{lcl} v_{ji}^{(t+1)} & = & v_{ji}^{(t)} + \eta_{\mathrm{input}} * \mathrm{sign}((y^{(t)} - \hat{y}^{(t)}) w_i^{(t)}) \\ & * \sigma_i(x^{(t)}) (1 - \sigma_i(x^{(t)})) x_i^{(t)} \end{array}$$

where $\sigma_i(x^{(t)}) = \frac{1}{1+e^{-(v_{ji}^{(t)}x_j^{(t)}-\beta_i^{(t)})}}$ is the logistic function activation of the *i*th feature, and the learning rate η_{input} was selected by tuning to be a constant value of 0.01.

In order to investigate how feature evolution synergizes with backpropagation, we conduct further experiments. First, we make the learning problem more difficult by increasing the number of hidden units in the target network, then we supplement the above learning methods by first backpropagating the error before performing feature selection or evolution.

Fig. 3 shows how the performance of the experimental setups vary when the number of hidden units in the target network is increased to either 100 or 500. The left side of these figures uses the experimental setups as described above where only the readout weights are adjusted via the Delta-rule, the right side of these figures additionally incorporate the modified backpropagation just described. These results are discussed below.

Discussion

Looking at the left hand plots of Fig. 3 we see that as the problem is made more difficult, the advantage conferred by feature evolution appears to decrease. While the best performances are still found in the "evolution" regime, it now requires 500 learnable features to outperform selection with 10,000 learnable features in the 100 hidden target case, and requires 10,000 learnable features to do so when there are 500 hidden units in the target network. This could be for several reasons: one possibility is that as the prediction problem becomes more complex the necessity of maintaining a diverse set of features becomes more pressing. The "selection" regime naturally accomplishes this by introducing new features that are unrelated to existing features. Finding effective, computationally inexpensive methods for promoting feature diversity in the "evolution" regime will require further work.

For these experiments, altering the learning procedure to include backpropagation of error drastically improves the performance of the "no selection" regime (Fig. 3, right hand

plots). This is essentially going from linear regression to the typical supervised ANN learning procedure. The performance of the "selection" regime is also either improved or left approximately the same. However, feature evolution no longer affects much improvement beyond purely selectionist methods.

This last point may be due to how feature weights are replicated. Here, the genome of a feature is considered to be its initial input weight vector, not the weight vector that has been altered through backpropagation—the evolutionary process is Darwinian rather than Lamarckian. So, any useful feature learning that arises as a result of backpropagation is not inherited through reproduction, and newly born features must then learn the backpropagated weight changes anew—just as under the selectionist method. Lamarckian evolution is known to be unstable in dynamic environments over phylogenetic timescales (Sasaki and Tokoro, 1997), but whether Lamarckian evolutionary neurodynamics combined with backpropagation also falls victim to the same pathologies remains to be tested in future work.

Conclusion

In this work we have introduced a method (Online Extreme Evolutionary Learning Machines) for automatically searching for features in an online learning task through feature replication. This process essentially makes the features (ANN hidden nodes) the members of a population undergoing an online, steady state evolutionary algorithm. We have demonstrated that using OEELMs results in significantly better predictive accuracy as compared to either using a fixed set of features or a purely selectionist search method, for both stationary and non-stationary environments.

Additionally, we have explored feature evolution as it relates to learning features through the backpropagation of error. Here, feature evolution is capable of achieving similar or better error than backpropagation in many instances (see Fig. 3). This result alone is interesting, because feature evolution may be more widely applicable than backpropagation: it does not require having known, differentiable features, but in principle could be used to evolve features of any form. Another natural next step is the application of evolution to convolutional neural networks (LeCun and Bengio, 1995). Here instead of hand-designing the shapes of the kernels that produce feature maps, the shapes and properties of kernels themselves can be evolved. Finally, it will be worthwhile to investigate how encoding features with a more evolvable, indirect encoding such as HyperNEAT (Stanley et al., 2009) may improve performance even further.

Feature evolution, as inspired by the Neuronal Replicator Hypothesis, is a promising method for online learning tasks. We foresee it being of particular relevance for robotics applications, where robots must learn predictive models in order to operate in unknown and possibly non-stationary environments. Specifically, we foresee these methods be-

ing useful for a robot to learn forward and inverse models (Wolpert and Kawato, 1998) from which it may fantasize against in order to accomplish some task in a given environment. This should allow for a robot to easily adapt to damage (Bongard et al., 2006) or to changes in morphology brought about by evolution, development or self-reconfiguration. This promising area will be investigated in future work.

Source Code

The source code used for all experiments conducted in this paper is available online at https://github.com/jauerb/OEELM.

Acknowledgements

The authors thank Rupam Mahmood for his kind cooperation and advice.

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 308943 and the "Bayes, Darwin, and Hebb" Templeton Foundation FQEB Grant.

References

- Arthur, W. B. (1994). Inductive reasoning and bounded rationality. *American Economic Review*, 84(2):406–411.
- Becker, S. T. S. and Obermayer, K., editors (2003). *Adaptive non-linear system identification with echo state networks*. MIT Press Cambridge, MA.
- Bongard, J. and Lipson, H. (2007). Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Science*, 104(24):9943–9948.
- Bongard, J., Zykov, V., and Lipson, H. (2006). Resilient machines through continuous self-modeling. *Science*, 314:1118–1121.
- Ciresan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural computation*, 22(12):3207–3220.
- Clark, A. (2013). Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, 36(03):181–204.
- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.
- Enns, R. H. (2010). It's a Nonlinear World. Springer.
- Fernando, C., Goldstein, R., and Szathmáry, E. (2010). The neuronal replicator hypothesis. *Neural computation*, 22(11):2809–2857.
- Fernando, C. T., Szathmary, E., and Husbands, P. (2012). Selectionist and evolutionary approaches to brain function: a critical appraisal. *Frontiers in Computational Neuroscience*, 6(24).

- Germann, J., Auerbach, J., and Floreano, D. (2014). Programmable self-assembly with chained soft modules: an algorithm to fold into any 2-d shape. In *Proceedings of the International Conference on the Simulation of Adaptive Behavior*. To Appear.
- Hawkins, J. and Blakeslee, S. (2007). On Intelligence. Macmillan.
- Helmholtz, H. v. (1860). Handbuch der physiologischen optik, vol. & trans. jpc southall.
- Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.
- Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257.
- Huang, G.-B., Zhou, H., Ding, X., and Zhang, R. (2012). Extreme learning machine for regression and multiclass classification. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 42(2):513–529.
- LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361.
- LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560.
- Mahmood, A. R. (2014). Personal Communication.
- Mahmood, A. R. and Sutton, R. S. (2013). Representation search through generate and test. In *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). *Learning representations by back-propagating errors*. MIT Press, Cambridge, MA, USA.
- Sasaki, T. and Tokoro, M. (1997). Adaptation toward changing environments: Why darwinian in nature. In *Fourth European conference on artificial life*, pages 145–153. MIT Press.
- Schmidhuber, J., Wierstra, D., Gagliolo, M., and Gomez, F. (2007). Training recurrent networks by evolino. *Neural computation*, 19(3):757–779.
- Stanley, K., D'Ambrosio, D., and Gauci, J. (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212.
- Sutton, R. S. and Whitehead, S. D. e. a. (1993). Online learning with random representations. In *ICML*, pages 314–321. Citeseer.
- Widrow, B. and Hoff, M. E. e. a. (1960). Adaptive switching circuits. In *IRE WESCON Conv. Rec.*, volume 4, pages 96–104. Defense Technical Information Center.
- Wolpert, D. M. and Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7):1317–1329.

Berry Eaters: Learning Colour Concepts with Template Based Evolution

Chris Headleand¹, Llyr Ap Cynedd¹ and William J. Teahan¹

¹Bangor University, Wales, UK c.headleand@bangor.ac.uk

Abstract

In this paper, we propose an approach inspired by Gärdenfors conceptual spaces, to represent concepts (specifically colour) in a situated embodied agent. Through this we also extend the work of Braitenberg by proposing an alternate vehicle capable of recognising colour concepts. Furthermore, we demonstrate how it can learn to distinguish between different colours by evolving our vehicle's conceptual space using Template Based Evolution.

Background and motivation

Categorisation and the ability to recognise and represent concepts has been described as a fundamental cognitive ability and a primary research goal of AI (Gärdenfors and Williams, 2001). The traditional approach from classical AI is to represent concepts symbolically using a top-down design process. The bottom-up subsymbolic (connectionist) approach, in contrast, emphasizes the importance of symbol grounding and the crucial role that the body has to play, by having the agent situated in the world and embodied—able to perceive and interact with the world through sensorymotor co-ordination (Pfeifer and Scheier, 2001).

Being able to recognise similarity is key to many biological phenomena, such as how simplistic agents are able to learn what food is safe to eat. Whilst prototypical items are easily represented through the symbolic approach, this does not take into account possible variations that naturally occur in real-life. Take the problem of colour recognition, for example. In natural systems, it is very unlikely to find exactly the same colour—no leaf on a tree is ever the same shade of green, or a berry is ever the same shade of red.

Gärdenfors (2004) presents an alternative methodology, called Conceptual Spaces which models knowledge in the form of regions in a geometric space made up of quality dimensions. By modeling in this way, Gärdenfors argues that an agent is able to recognise variations on a defined concept prototype. A primary motivation for this paper is to see if a Conceptual Spaces representation can be implemented in a situated, embodied agent in a manner analogous to the approach adopted by Pfeifer and Scheier (2001) in order to

demonstrate how concepts such as colour can be learnt in an unsupervised way.

Conceptual Spaces

In his book, Gärdenfors (2004) proposes Conceptual Spaces as an alternative and possible bridge between alternative knowledge representation methods, specifically the symbolic and connectionist approaches. One of the principle problems this method seeks to address is the modelling of abstract concepts. Concepts, as a theory within cognitive science, focus on the understanding of similarity between entities, allowing for categorisation. In Gärdenfors' theory, a Conceptual Space is a representation model where each concept is represented by a region in a geometric space defined by a number of quality dimensions. For example, if we were to represent fruit, we may consider the size, shape, taste and texture as different quality dimensions. Colour is a good example as we naturally represent it dimensionally; Gärdenfors discusses the HSV (Hue, Saturation, Brightness) colour space, as it represents the phenomenal structure of the colours rather than the scientific properties. There is no claim made that this closely represents human perception; in fact there are many other dimensionally represented colour spaces which could also be applied to the same representation including RG, RGB, CMYK and NCS Wyszecki and Stiles (1982). Interestingly the eye has detection mechanisms along three distinct spectrums (Blue-violet, Green and Yellow-Red) so even at the detection level, we perceive colour dimensionally.

Braitenberg Vehicles

Braitenberg (1986) proposed a series of hypothetical thought experiments concerning what he called "vehicles". These self operating machines were designed to show how seemingly complex behaviour that an observer agent might associate with traits such as fear and aggression could be exhibited from relatively simple sensory motor connections. We will now discuss vehicle types 2 to 4 and 7 in particular as they directly relate and have inspired our approach.

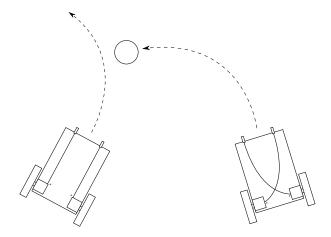


Figure 1: The direct connection type 2a vehicle turns away from the light, where as the type 2b turns towards the light.

Types 2 and 3 A type 2 vehicle has two light sensors connected to two motors. Type 2a has a direct connection (the left sensor to the left motor and right to right) while in the 2b model the connections are crossed over. This type of design allows the vehicles to demonstrate rudimentary fear (2a) and aggression (2b) towards a light source as they will either turn towards or away from it (see figure 1). With the type 3 model, the connections are negative (inhibitory) as opposed to positive (excitatory); for example, a light source will slow a motor rather than speed it up. This produces a more gradual effect, vehicles that circle a light source, or vehicles that will stay close to a light source while exploring their environment.

The type 3c vehicle has a mix of sensors with a variety of connections allowing the vehicle to display all the characteristics of the other type 2 and type 3 vehicles. These vehicles are described as having light, heat, oxygen and bio matter sensors connected respectively in the aggressive, fear, loving and exploring formats. This type of vehicle is able to produce complex behaviour, such as appearing to show preferences, disliking certain areas and favouring others.

Types 4 and 5 In the type 4 and 5 vehicles, threshold devices are introduced to the design. These pass-through devices limit power by allowing no output until a threshold level has been reached, or inversely by allowing everything until a threshold has been reached. In type 4 vehicles, this principle is used to simulate instinctive decisions, advancing the base type 3 vehicles. In type 5 vehicles, Braitenberg discusses how these thresholds could be used to develop a simple logic system and memory.

Concepts In the type 7 vehicle, Braitenberg discusses how concepts can be represented in a vehicle using a hypothetical component called Mnemotrix. This component, described as a wire, has an interesting property—its resistance is high

until an electrical current simultaneously traverses the two components it is connected between. This allows the vehicles to associate input with a specific behaviour. For example, if the oxygen sensor was high while the bio-sensor was also activated, the vehicle can "learn" to associate biology with oxygen by, quite literally, crossing its wires. While this form of association is one definition of a concept (a general idea derived from instances or occurrences), it does not describe the abstract concepts this paper seeks to explore.

Applications

Braitenberg vehicles have been applied to a large number of experimental and practical applications, particularly as models for animal behaviour, but they have also been directly applied to robotics. For example, Yang et al. (2006) applies Braitenberg vehicles to the navigation of mobile robots and Lilienthal and Duckett (2003) experiments with using types 2, 3 and 4 Braitenberg vehicles to allow a mobile robot to locate a static odour. Salumae et al. (2012) applied the type 2b controller to the design of an underwater robot, with sensors allowing it to localise and navigate a current. Braitenberg vehicles, due to the simplicity of their design, provide an exceptional platform for experimental investigation into a variety of reactive embodied behaviours.

Fungus Eaters

In 1962, Toda described a thought experiment involving autonomous agents known as fungus eaters (Pfeifer, 1996; Toda, 1962). This study was the origin of the concept of complete autonomous agents and one of the earliest works in embodied artificial intelligence, pre-dating Brooks by almost 30 years.

Toda describes a robot "sent to a hypothetical planet as a robot uranium miner, which sustains itself by eating fungi as its energy source". The solitary fungus eater has a means of collection and locomotion and a means of making decisions while interacting with its environment. It must be autonomous as it cannot receive any external aid.

Toda's original argument discusses how notions of emotion, irrationality and social behaviour can be applied to a Fungus Eater simply as a function of operating within its task environment and attempting to maintain itself by eating fungi.

Template Based Evolution

Template Based Evolution (TBE) is a methodology for evolving reactive, subsumption based agents. This is achieved implicitly, by testing agents through trials in an environment, rather than via a fitness function. The motivation behind the development of the algorithm and methodology was the work of John Holland (1990).

Genome

The Genome in a TBE simulation is a collection of unique attributes, each attribute taking a specific role in the behaviour of the agent. In the original paper (Headleand and Teahan, 2013), two attribute types were defined, action and trait which are usually represented numerically. This was subsequently extended in (Headleand, 2013) to include threshold attributes.

Trait Trait attributes represent qualities of the agent, for example, colour, speed, energy, weight etc. They can be independent of other attributes but can also affect selected behaviours; for example, a trait of speed may affect a run behaviour.

Action An action attribute is an index value which represents the evolved selection of an action within a list of possible actions within the subsumption. During each time step, as the agent moves through its subsumption, the output is determined by the action attribute.

Threshold The third class of attribute described in Headleand (2013) was for alternate styles of simulation. This is used to define the threshold where different higher layers of the template subsume lower layers. Alternatively, it may be used to define activation functions or decision processed within specific actions.

Design of a TBE simulation

A TBE simulation is built from the bottom up, starting with the environment and ending with the individual agents. This formal methodology forces the designer to consider the ecological niche of the agents. As a result, the task environment is integral to the behaviour rather than an abstract container, which the agents are simply placed within.

Environment The environment is the domain where each agent's suitability is tested through successive trials. A trial is a challenge or obstacle specific to that environment, which may directly affect the agent's ability to survive (and subsequently reproduce). Trials can take multiple forms, for example if we were to simulate the behavioural evolution of herd animals, a trial may be a predator, exposure to adverse weather or the availability of food.

Species All agents of a species of a specific species must conform to a basic prototype. This includes the species template and the genome prototype (the amount and type of attributes). The species template is a subsumption architecture, with either outputs or conditions being determined by attributes from the agents genome. This ensures that whilst the subsumption architecture template is common across all agents in the simulation, the agents' unique genome will define their behavioural responses. (For an example, refer to figure 3).

Agents As the subsumption template is established within the species definition, each unique agent is only required to maintain their own genome. This allows for a moderate efficiency increase over alternative methods such as evolving an Artificial Neural Network or Markov Network where each unique agent in the simulation requires the storing of a full network.

Applications

In Headleand and Teahan (2013), the TBE methodology was applied to simulating the behavioural evolution of migratory birds. The purpose of this study was to test the methodology but also to assess whether the predator escape tactics proposed in the literature could be evolved implicitly. The study demonstrated that the TBE methodology was capable of evolving complex behavioural and morphological changes such as wide vision and group escape tactics.

Berry Eaters: An adapted thought experiment

In the background to this research, we described two thought experiments, Toda's fungus eaters and Braitenberg vehicles. In this section, we will discuss an adaptation that adopts aspects of each—the Berry Eaters (BE) experiment has been devised to allow exploration of abstract concept learning within reactive embodied agents.

The Scenario

As discussed in Toda's "the Design of a Fungus Eater" (Toda, 1962), the author describes a hypothetical scenario where robots are sent to a planet where a native fungus is used as fuel. We propose an alternative to this experiment where there are two (rather than one) similar fuel source options. One fuel source will power the robot (known as edible) the other is incompatible with the robot (known as poisonous). These two alternatives are distinguished by one key feature—colour—for the purpose of this study. To add additional complexity, the designers of the eaters will have no knowledge of these features during the design process.

In order to differentiate our thought experiment between this and the fungus eaters experiment, the fuel source will be berries.

Design of a Berry Eater

The Braitenberg vehicles demonstrate how seemingly complex behaviours can emerge from simple sensory motor couplings. The insight gained from these experiments provides us with possible theories as to how simple creatures, such as ants, can demonstrate complex behaviour despite limited cognitive capacity.

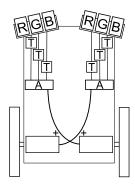
As proposed for our experiment, there are two types of berries in this experiment, edible and poisonous. For the purpose of discussion, let's say that the edible berries are red and the poisonous berries are blue. On face value, recognition in this case seems relatively straight forward. However, consider the colour red. Although we may have a clear idea of what constitutes a prototypical red, in fact there are many of different shades of red due to slight variations in the hue, saturation and brightness. When we consider these variations, we are faced with boundary problems such as defining the point that red stops and pink begins, for example. As humans who can easily recognize colour differences, clearly we have an abstract conceptual understanding that allows us to overcome these problems. It is also reasonable to assume that this ability is adaptive, developing as new information becomes available.

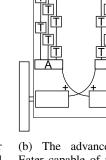
Basic Design – BE Type 1 Using a Brightenberg vehicle as a basis for the design, we can combine the qualities of a type 2a and 2b vehicle to create a hybrid that would turn away from one artefact and towards another. However, this only considers the sensory motor coordination; it does not address in itself how such a vehicle could learn which colour of berry is edible and which is poisonous, or how to go about detecting colour in the first place.

The problem of how to detect colour can be overcome in the following way (we call this a BE Type 1). Each light sensor in our hybrid vehicle is replaced with three wavelength sensors either side calibrated to red, green and blue respectively. As the sensor detects a wavelength in its calibrated spectrum, it produces an electrical signal. A circuit with a preset threshold and cut-off (such as within the type 5 vehicles) can be used to only produce an electrical signal at a particular value along that spectrum. We can take the three outputs from these threshold components into a set of switches, allowing current to pass through only if all three threshold devices are triggered simultaneously (for discussion's sake, lets call this component an activator). The combination of these components allows us, as the designer, to set the BE to respond to a specific colour (see figure 2.(a) where we have illustrated the circuitry required to get the BE to move towards a colour). Furthermore, we can include two sets of activators, one for a specific colour to avoid and another for a colour to be attracted to.

This would result in a Berry Eater that can move towards and away from two different coloured objects. However, this requires that the objects match an exact colour. This current configuration does not allow for the variation in the colour of the berries we discussed earlier; even an altering of the light conditions would render this particular design of berry eater incapable of operation.

Advanced Design – BE Type 2 Our more advanced design of a berry eater (BE Type 2) is capable of recognising variation in colour by incorporating a conceptual spaces inspired model of representation. In the previous BE Type 1 design, we had a single threshold/cut-off circuit. However, for our more advanced design, we propose using two circuits, a threshold that only allows signals to pass through





(a) The basic Berry Eater design, equipped to respond to a single colour. A duplication of these components with the wires wired directly instead of crossed over would allow the BE to move away from another colour.

(b) The advanced Berry Eater capable of representing colour concepts. Notice the threshold component triggered only if a signal is above a specific value and the cut-off component that allows all signals below a specific threshold.

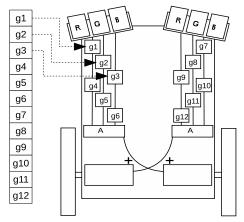
Figure 2: Design of a Berry Eater.

if they are over a certain value followed by a separate cutoff that only allows signals below a certain value to pass through. Using this system, it is possible to define a RGB conceptual space type controller that is able to encode variations around a colour prototype in order to represent colour concepts; see figure 2.(b). In addition, through the unsupervised evolutionary process described below, the agent is able to learn the colour of the two distinct classes of berries, those that provide fuel and those that will destroy it.

For the simulation of our final Berry Eater design, we first generate a population of Berry Eaters with randomly assigned threshold values and deploy the agents within the environment. Before new BE's are manufactured, the threshold values of two surviving agents (the parents) are recombined and mutated to produce the thresholds of a new agent (the offspring). To help ensure that the parents are successful agents, we only breed from two parents who have survived for a given amount of time (a maturity measure). The agents unable to find edible berries will have run out of power (due to starvation), and agents unable to differentiate between edible and poisonous berries will also have been destroyed, ruling them out for reproduction.

The survival and success of many species relies on the interaction of multiple agents to produce new offspring. However, the BE design does not contain any mechanisms for communication other than breeding and each BE does not even have the ability to sense each other's presence. While we can predict the possible behaviour of an individual BE, it is significantly harder to imagine what emergent qualities a colony of berry eaters may exhibit. To explore this question, we performed the simulation as described below.

Genome to Berry Eater Phenome Mapping



TBE Template (left sensor)

```
sR = sensor-Red;
sG = sensor-Green;
sB = sensor-Blue;
R,G,B = false;

//check sensor reading is within
//conceptual space defined by genome
if(sR > g1 && sR < g4) {R = true}
if(sG > g2 && sG < g5) {G = true}
if(sB > g3 && sB < g6) {B = true}

//output to motor
if( R&& G && B){set right-motor + 1}
else {set right-motor - 1}
```

Figure 3: An illustration of the genome attribute to BE phenome mapping for the "Box Representation" approach, allowing the Berry Eater to be attracted towards a colour, and pseudo-code implementation of the left sensor template. Note that these components would need to be duplicated with the motor wires uncrossed for the Berry Eater to also avoid a colour.

Experimental Results

In this section, we will discuss the design of the Berry Eaters simulation and the experimental results. The design focuses on the TBE methodology, starting with the environment and ending with the individual agents. The purpose of this study was to explore two principle research questions:

- 1. Could a Conceptual Spaces inspired approach be implemented within a reactive embodied agent to provide a mechanism for representing colour concepts?
- 2. What behaviours would a group of Berry Eaters produce when placed into an environment?

Environment

The environment that we simulated in the NetLogo programming language is a wrapped world 50 units high by 50 units wide. Each unit, known as a patch in NetLogo, is large

enough to fit a berry in its entirety, though multiple berries may be stacked on top of each other.

Within the environment, there are the poisonous berries described previously, and any agent who eats a poisonous berry is destroyed. All agents will starve if they are unable to eat within a specific time period (set at 360 ticks within the simulation). To be able to breed, all agents will also have to reach a maturity which is set to 400 ticks within the simulation requiring breeding agents to have eaten a berry. These values were selected due to the size of the environment, 360 ticks being considered enough opportunity to find a food source. 400 ticks was selected as it forced a BE to eat at least once before it was able to breed (to exclude unsuccessful agents).

Species

In the previous section, we presented a Braitenberg-style design of the Berry Eater. This design is implemented as a subsumption architecture within the species template (see figure 3). In the pseudo-code shown in the figure for the left sensor (attraction), the RGB readings are checked to see if the detected colour falls within the conceptual space defined by the genome attributes, if the do the right motor speed is increased. For this study the population size was fixed to 500 vehicles, each new vehicle had a mutation chance set to 5%.

The attributes within the genome definition represent the threshold values for the colour detection. This requires six attributes for positive (edible) detection and a further six attributes for negative detection (poison).

Agent

The agents are Berry Eaters that maintain their own unique genome of attributes. The agents respond to their environment based on the signals they receive from their virtual sensors. This is illustrated in figure 3. If the agents receive no signals, they will simply move forward, wandering slightly to simulate friction/slip of their wheels on the ground as with traditional Braitenberg Vehicles.

Attributes For the purpose of this study, threshold attributes have been used to define the conceptual spaces as described in an earlier section on Template Based Evolution and in (Headleand, 2013).

Representation The BE Type 2 design described one method of defining a conceptual space, using upper and lower thresholds along each dimension to construct a region in 3D space. In this section, we discuss how this box is implemented and present an alternative, sphere-based representation which uses a single point in RGB space, and a radius which defines the region.

Box representation: The box representation uses a lower and upper threshold to define three acceptable ranges along three

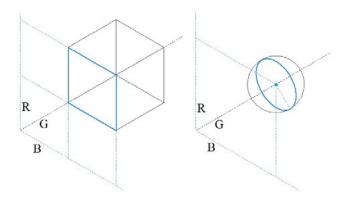


Figure 4: An illustration of two conceptual regions defined around their central colour prototypes, for both the box and spherical representations.

dimensions, creating a box in 3D space. This method is similar to the original conceptual spaces theory proposed by Peter Gärdenfors. In this method, any color which falls within this region is considered conceptually similar to the learned prototype. To implement the box representation in TBE, six threshold attributes were used to define a shape in three dimensional RGB space.

Spherical representation: An alternative to the box representation was also implemented. In this version, a single point in 3D space, the colour prototype, was learnt by the Berry Eater. In addition, the Berry Eater also learnt a single value representing an acceptable distance from the prototype, creating a sphere in 3D space. In this representation, any colour which fell within this sphere was considered conceptually similar to the learnt prototype. To implement this method in TBE, three threshold attributes were used to represent the position within the space and a fourth attribute represented the distance of the radius of the sphere.

Colour Space In this study, we have chosen to represent the RGB colour space. Although HSV more closely matches Gärdenfors original description, adopting RGB has the advantage that RGB sensors are readily available, and therefore real-world Berry Eaters can be developed in future to see how the simulated and real-world equivalents compare.

Experimental Setup

We conducted several experiments in order to help answer the two research questions posed above. Six different experimental setups were designed (detailed in Table 1), and for each of these experiments, we implemented both the box and the spherical conceptual space representation (experiment variants (a) and (b)). For all experimental setups (12 in total), we ran 100 simulations (1200 simulations in total). Each simulation was allowed to run for 5000 ticks, at which point the results were recorded.

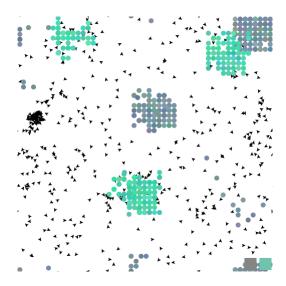


Figure 5: Berry eaters demonstrating feeding frenzy behaviour: in the middle of the left side of the image, we can see a feeding frenzy about to reach its conclusion. At the edible (green) clump towards the bottom of the centre of the image, we can see another beginning to form.

Group Behaviour Patterns

Various behaviours were visibly apparent during the simulations as discussed below. Although the agents have no knowledge of each other, the simple environment to agent feedback has resulted in visually complex group behaviours.

In this section, we have made no differentiation between which conceptual representation was used (sphere or geometric) as this did not make any difference in the behavioural outcome.

Feeding Frenzy When the berry-growing algorithm was set to generate a clump of berries, the BEs in the simulation regularly demonstrated a feeding-frenzy behaviour. This is where the vehicles would approach from multiple directions, swarming around a particular food source. The swarm would get larger and denser as the berries were eaten and re-grown in different locations within the clump, until eventually all the berries were eaten and a clump formed elsewhere.

In the middle of the left side of figure 5, we can see the final results of a feeding frenzy, with agents in a dense swarm covering a diminishing edible clump. Additionally in the center of the image towards the bottom, we can see a second beginning to form, with agents beginning to swarm from above and left.

Path Following If the berry growth algorithm was set to random, the BEs would end up moving along similar paths. These paths resemble ant trails, and therefore provide the impression that the agents are performing a "follow the leader" type behaviour. However, the agents have simply

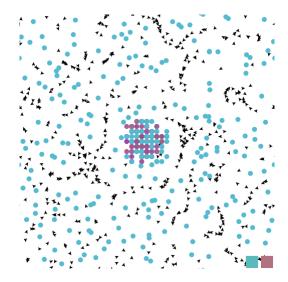


Figure 6: Berry eaters demonstrating path following behaviour: notice how the agents in the simulation appear to be following each other on a path between the poisonous berries. Around the clump in the centre, we can observe the apprehensive behaviour mentioned later in section .

learnt the correct colours and are navigating the environment by being repelled away from the poisonous berries and being attracted to the edible ones. This results in reactive pathfollowing behaviour based on the placement of the berries in the simulation (see figure 6).

Pseudo-Flocking Often the agents would demonstrate a behaviour that resembled flocking. Indeed, this flock would move in unison for a significant amount of time before being broken up by the presence of food or poisonous berries to avoid. A group of agents would be attracted towards a series of berries (often forming a path). When the available berries were eaten, the group of agents would begin to align. As they were subsequently repelled or attracted to other berries along their route, the flock would maintain its structure until a large food source or clump of poisonous berries forced the flock to disperse.

Apprehension If the berry growth algorithm was set to clump, occasionally the poisonous and edible berries would grow together. This caused interesting apprehensive-like group behaviours to appear as a result. The BEs would approach from the outside, but once there they would twitch and turn in multiple directions. Occasionally one would rush forward before quickly returning back to the outside. In figure 7, you can see that the berry eating trend has begun to taper at the top of the graph. There is also a significant step a little lower. Both these drops in the berries being eaten have occurred because of the described behaviour.

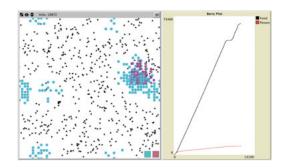


Figure 7: Berry Eaters demonstrating apprehensive behaviour: on the graph, we can see one step where this situation has already taken place and at the top of the graph we can see another one beginning to form. A few agents can be seen exploiting a few solitary edible berries that have grown between the poisonous clumps. However, the rest of the colony has avoided the clump creating a small exclusion zone around the area.

Colour Concept Acquisition

The problem of colour concept acquisition is the core focus of this study. As described above, two possible methods of colour conceptual space representation have been explored: the first is where the concept is represented geometrically in 3D space as a sphere (or as a hypersphere in higher dimensions), and the second is where the concept is represented in 3D space as a box. We also discussed two berry growing algorithms, one which caused the berries to grow in clumps and another which caused the berries to grow randomly within the environment.

A colour has been considered 'successfully learnt' if the average colour that the agents have learnt has a Euclidean distance of less than some threshold distance from the colour prototype which is randomly generated at the beginning of the experiment. (In our simulation, we set the threshold to 20 because two colours with a Euclidean distance of less than 20 were visually of the same prototypical colour for a human observer).

The 12 different experimental setups are presented in table 1. In the second column, we have stated which berry growing algorithm was used: either Clump or Random. The column labelled 'Rep.' details the concept representation method that was used—either box, or spherical. In the columns labelled 'E' and 'P', we detail the number of edible ('E') and poisonous ('P') berries in the simulation. In the column labelled 'LE', we list the number of unique simulations where the agents successfully learnt the colour of the edible berries. In the column labelled "LP", we list the percentage of experiments where the agents successfully learnt the colour of the poisonous berries.

Setup	Growth	Rep.	Е	P	LE	LP
1.(a)	Clump	Box	300	300	71	62
1.(b)	Clump	Sphere	300	300	75	64
2.(a)	Random	Box	300	300	61	56
2.(b)	Random	Sphere	300	300	73	45
3.(a)	Clump	Box	150	450	94	87
3.(b)	Clump	Sphere	150	450	96	99
4.(a)	Random	Box	150	450	71	63
4.(b)	Random	Sphere	150	450	85	92
5.(a)	Clump	Box	450	150	63	48
5.(b)	Clump	Sphere	450	150	67	40
6.(a)	Random	Box	450	150	97	22
6.(b)	Random	Sphere	450	150	78	11

Table 1: Results from various setups of the Berry Eaters simulation.

Analysis

From table 1, we can see that the agents were typically successful in learning the colours in the simulation. Also, typically the spherical representation produced superior results. However, setups 6.(a) and 6.(b) produced confusing results: despite the fact that the environment was still populated with poisonous berries, the agents seemingly did not learn the poisonous colour although observations of these experimental setups showed that the agents still avoided poisonous berries.

For setup 6.(b) (spherical representation), we found that the agents developed an interesting coping mechanism. Instead of learning the specific poisonous colour prototype, the BEs learnt to avoid large but inaccurate conceptual space, however due to the size of this region it typically included the poisonous colour.

Conclusion

In this paper, we have discussed using a geometric representation as a viable method of representing concepts. This was demonstrated through the Berry Eaters experiment where virtual agents were able to evolve a simple conceptual understanding of colour which directly influenced their behaviour. We explored how these artificial life forms could be built following the same format proposed by Braitenberg and tested them through simulation.

Despite having no awareness of each other, the Berry Eaters acted in such a way that gave the appearance of group behaviour simply by following common rules and reacting to the environment. The simplest form of this pseudogroup behaviour was agents moving in the same direction, either along paths or in clusters. However, we also observed flocking-like behaviour, which exhibited many of the emergent qualities visible in more complex models such as Boids (Reynolds, 1987). This type of behaviour is usually only observed in models where the agents are aware of each other;

however, our simulation demonstrates that an environment based approach could generate comparable behaviour.

Typically, the spherical representation was more successful in achieving the goal of colour concept learning, although there was evidence of over-fitting and this approach is also further from Gärdenfors original theoretical framework. Future studies will explore other methods of geometric representation to determine whether further improvements can be gained.

Acknowledgements

Chris Headleand would like to thank Fujitsu for their ongoing financial support.

References

- Braitenberg, V. (1986). Vehicles: Experiments in synthetic psychology. MIT press.
- Gärdenfors, P. (2004). Conceptual spaces: The geometry of thought.
- Gärdenfors, P. and Williams, M.-A. (2001). Reasoning about categories in conceptual spaces. In *IJCAI*, volume 1, pages 385–392.
- Headleand, C. J. (2013). Bio-inspired methods of behaviour generation. Master's thesis.
- Headleand, C. J. and Teahan, W. J. (2013). Template based evolution. In Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion, pages 1383–1390. ACM.
- Holland, J. H. et al. (1990). Echo: Explorations of evolution in a minature world.
- Lilienthal, A. and Duckett, T. (2003). Experimental analysis of smelling braitenberg vehicles. environment, 5:10.
- Pfeifer, R. (1996). Building fungus eaters: Design principles of autonomous agents. *From animals to animats*, 4:3–12.
- Pfeifer, R. and Scheier, C. (2001). *Undertanding Intelligence*. MIT press.
- Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH Computer Graphics*, volume 21, pages 25–34. ACM.
- Salumae, T., Ranó, I., Akanyeti, O., and Kruusmaa, M. (2012). Against the flow: A braitenberg controller for a fish robot. In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pages 4210–4215. IEEE.
- Toda, M. (1962). The design of a fungus-eater: A model of human behavior in an unsophisticated environment. *Behavioral Science*, 7(2):164–183.
- Wyszecki, G. and Stiles, W. S. (1982). *Color science*, volume 8. Wiley New York.
- Yang, X., Patel, R. V., and Moallem, M. (2006). A fuzzy– braitenberg navigation strategy for differential drive mobile robots. *Journal of Intelligent and Robotic Systems*, 47(2):101–124.

Artificial Chemistries, Cellular Automata and Self-Organizing Systems

An Analog Chemical Circuit with Parallel-Accessible Delay Line for Learning Temporal Tasks

Peter Banda¹ and Christof Teuscher²

¹Department of Computer Science, Portland State University banda@pdx.edu
²Department of Electrical and Computer Engineering, Portland State University teuscher@pdx.edu

Abstract

Current synthetic chemical systems lack the ability to selfmodify and learn to solve desired tasks. In this paper we introduce a new parallel model of a chemical delay line, which stores past concentrations over time with minimal latency. To enable temporal processing, we integrate the delay line with our previously proposed analog chemical perceptron. We show that we can successfully train our new memory-enabled chemical learner on four non-trivial temporal tasks: the linear moving weighted average, the moving maximum, and two variants of the Nonlinear AutoRegressive Moving Average (NARMA). Our implementation is based on chemical reaction networks and follows mass-action and Michaelis-Menten kinetics. We show that despite a simple design and limited resources, a single chemical perceptron extended with memory of variable size achieves 93-99% accuracy on the above tasks. Our results present an important step toward actual biochemical systems that can learn and adapt. Such systems have applications in biomedical diagnosis and smart drug delivery.

Keywords

chemical delay line, chemical perceptron, chemical reaction network, analog asymmetric signal perceptron, temporal learning, chemical computing

Introduction

Learning and adaptation allow organisms to generalize and predict to the ever-changing environment they live in, which leads to a competitive advantage for their survival and reproduction. Artificial neural networks (Rojas, 1996; Haykin, 2009) are the prototypical example of artificial learning, however, their abstraction of biological systems is usually high. Building synthetic or actual molecular systems that can adapt and learn autonomously is a grand challenge for various reasons. First, the integration of the forwardpass (output production) with the backward-pass, where a teacher's action affects the concentrations of biomolecular species that define the system's functionality, requires complicated and timing-sensitive, both positive and negative catalytic feedback loops. Second, there is no natural and immediate mapping of neural networks to chemical systems. So far, only the input-weight integration has been successfully

implemented in wet chemistry (Bray, 1995; Mills et al., 1999; Kim et al., 2004; Qian et al., 2011).

In our previous work we proposed several binary and analog chemical perceptrons (Banda et al., 2013, 2014; Banda and Teuscher, 2014). These systems represented the very first simulated chemical systems capable of fully autonomous learning. We used chemical reaction networks (CRNs), which are unstructured macroscopic chemistries where the reactions of symbolic species follow mass-action and/or Michaelis-Menten kinetics. While our previous systems were able to solve simple tasks, the goal of this paper is to propose new chemical learning systems that can tackle much more complex tasks, in particular tasks with temporal dependencies. This challenge is motivated by practical computational as well as biomedical applications. For example, any form of continuous environmental sensing is typically defined temporally, where the target function output depends not only on the actual but also on the previous inputs.

In current chemical reaction networks, it is difficult to coherently store and retrieve values as we are used to in traditional computer architecture. Here, we propose a specific kind of memory widely used in computer and electrical engineering, a delay line (DL), which buffers the past inputs over a sliding window and presents them for reading (consumption) both sequentially but also as a parallel output. Our previous chemical DLs (Moles et al., 2014) lack the ability to scale up and provide solely a sequential access to the content. To address these issues we introduce a new DL called a parallel-accessible DL (PDL), which achieves optimal performance by employing wait queues and operates on the basis of two alternating signals (catalysts).

We then integrate a PDL with an analog asymmetric signal perceptron (AASP), a chemical perceptron trained by supervised learning. In our setup, a PDL feeds an underlying AASP with past input concentrations, and therefore allows to solve tasks defined as time series. We evaluated the performance of our new delayed AASP (AASP-DL) on four temporal tasks: the linear moving weighted average, the moving maximum, and two variants of the Nonlinear AutoRegressive Moving Average (NARMA). We also scale the

system's size to more than two cached inputs to study the effect of memory on learning.

Given the available technology, the AASP-DL may have applications in the area of medical diagnosis and smart medication (LaVan et al., 2003), such as temporal signal processing, monitoring and detection of specific concentration patterns produced, e.g., by cancer cells in a host. Our approach could potentially replace hard-coded solutions and would allow to reuse (retrain) chemical systems without redesigning them. We suggest that our model could be used as a system-level blue-print for actual wet biochemical (substrate-specific) implementations, for example by using DNA-strand displacement (Soloveichik et al., 2010; Zhang and Seelig, 2011) and deoxyribozymes (Stojanovic and Stefanovic, 2003; Liu et al., 2009).

Chemical Reaction Network

A chemical reaction network (CRN) (Espenson, 1995; Dittrich et al., 2001) represents an unstructured macroscopic artificial chemistry where the reactions of symbolic species prescribe the behavior of the system. The species are not assigned a molecular structure yet, therefore they serve as placeholders for hypothetical wet chemicals that react in the same way. Since the reaction tank is assumed to be well-stirred, the system's state does not contain any spatial information and is effectively reduced to a vector of species concentrations, which we treat as a dimensionless quantity.

The reaction rate defines the strength or speed of a reaction application prescribed by kinetics laws. The rate of an ordinary reaction $aS_1 + bS_2 \rightarrow P$ is defined by mass-action law (Espenson, 1995) as

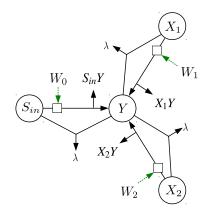
$$r = \frac{d[P]}{dt} = -\frac{1}{a}\frac{d[S_1]}{dt} = -\frac{1}{b}\frac{d[S_2]}{dt} = k[S_1]^a[S_2]^b,$$

where $k \in \mathbb{R}^+$ is a reaction rate constant, a and b are stoichiometric constants, $[S_1]$ and $[S_2]$ are concentrations of reactants (substrates) S_1 and S_2 , and [P] is a concentration of product P.

The rate of a catalytic reaction $S \xrightarrow{E} P$, where a substrate S transforms to a product P with a catalyst E, whose concentration increases the reaction rate, is given by Michaelis-Menten kinetics (Copeland, 2002) as

$$r = \frac{d[P]}{dt} = \frac{k_{cat}[E][S]}{K_m + [S]},$$

where $k_{cat}, K_m \in \mathbb{R}^+$ are rate constants. By combining kinetic expressions for all species, we obtain a system of ODEs that we simulate with a 4^{th} order Runge-Kutta numerical integration with step size 0.1.



(a) input-weight integration

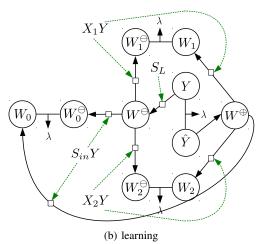


Figure 1: (a) The AASP's reactions performing input-weight integration, where cross-weight competition is achieved by the annihilation of the inputs S_{in}, X_1 , and X_2 with the output Y. Three species $S_{in}Y, X_1Y$, and X_2Y represent the contributions of the inputs S_{in}, X_1 , and X_2 with associated weights in the output Y. (b) The AASP's reactions responsible for learning: comparison of the output Y with the target-output \hat{Y} by annihilation $Y+\hat{Y}\to\lambda$, and subsequent positive and negative adaptation of the weights W_0, W_1 , and W_2 . Nodes represent species, solid lines are reactions, dashed lines are catalysts, and λ stands for no or inert species.

Analog Asymmetric Signal Perceptron

In this section we briefly describe the analog asymmetric signal perceptron (AASP) introduced in (Banda and Teuscher, 2014). The AASP is a CRN consisting of 17 species and 18 reactions that mimics a formal two-input analog perceptron (Rosenblatt, 1958). The input species X_1 and X_2 correspond to the formal inputs x_1 and x_2 , the species Y to the output y, and the input signal S_{in} to the constant coefficient x_0 of the bias weight w_0 . The AASP represents the weights by species W_0, W_1 , and W_2 . We optimized the

AASP's rate constants to tune its performance by genetic algorithms.

During the nonlinear input-weight integration (Figure 1(a)) each weight catalyzes a transformation of the input X_i to the output Y, and races with its input's annihilation $X_i + Y \to \lambda$ and the other weights since the output Y is shared. Naturally the output concentration cannot exceed all the inputs provided. When the weight concentration is close to zero, its branch could consume more from the output than it contributes, hence a low weight concentration could impose a negative pressure on a different weight branch. Therefore, a weight species coupled with its annihilatory reaction can act effectively as a catalyst and inhibitor.

The AASP is trained by classical supervised learning (Rojas, 1996) triggered by an injection of the target output \hat{Y} provided some time after the injection of the input species. The AASP compares the output Y and the target output Y by a rapid annihilation. A leftover of the output Y implies that the weights need to be decreased, hence a negative weight changer W^{\ominus} is produced from Y. In the opposite case, \hat{Y} transforms to a positive weight changer W^{\oplus} to increase the weights. For the positive adaptation, W^{\oplus} is distributed by concurrent catalytic reactions $W^{\oplus} \to W_i$ among the weights proportionally to their input-weight contributions $S_{in}Y, X_1Y$, and X_2Y . Similarly, the negative adaptation splits W^{\ominus} to intermediates $W_0^{\ominus}, W_1^{\ominus}$, and W_2^{\ominus} , which annihilate with the weights. Note that $S_{in}Y, X_1Y$, and X_2Y are three auxiliary species that represent the contributions of the inputs S_{in}, X_1 , and X_2 with associated weights in the output Y. Because of the annihilatory reactions, $[Y] \leq [S_{in}Y] + [X_1Y] + [X_2Y]$. Also, each learning iteration the input-weight contributions species are flushed.

Parallel-Accessible Delay Line

A chemical delay line stores past input concentrations in a queue and provide them to a connected system. We previously introduced two variants of a chemical DL (Moles et al., 2014). The core copy reaction $X_i \xrightarrow{X_i^S} X_{i+1} + X_{i+1}^C$ controlled by the signal (catalyst) X_i^S splits the cached input X_i (or injected input X) into one copy X_{i+1} , which is available for consumption, and another copy X_{i+1}^C , which is buffered and propagated to the next stage.

The manual signalling variant is fully sequential and relies on injections of signals X_i^S for each copy phase in a backward order, as shown in Figure 2(a). First, an injection of the X_3^S signal produces X_3 from cached X_2^C , then the X_2^S signal copies the cached X_1^C into X_2 and X_2^C , etc. Even though this model is error-free because of its sequential operation and a separation of the copying stages, it requires a significant amount of external "help" since the number of injections grows linearly with its size. The signal backpropagation DL keeps all the reactions from the manual version, but adds the transformations of signals $X_i^S \to X_{i-1}^S$, which

replaces the signal injections (Figure 2(b)). Because the backpropagation reactions are not instantaneous, there is always an overlap of consecutive stages, where both signals X_i^S and X_{i-1}^S are present. That produces an error that accumulates with every stage. Therefore, this model could be used only for small sizes, such as n=2 or n=3.

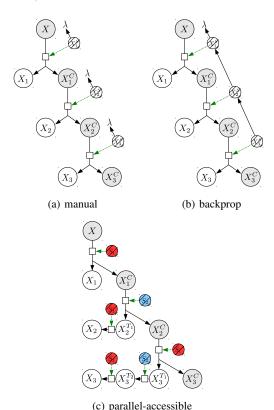


Figure 2: Three variants of a chemical DL of size n=3. The species X_1, X_2 , and X_3 represent the input X cached at time/cycle t, t-1, and t-2 respectively. The manual and backpropagation signalling models are sequential, hence they produce the cached values one after another. The parallel model with a wait queue X_i^T for each stage and two alternating signals S_1 and S_2 produces all the values simultaneously. Nodes represent species, solid lines are reactions, dashed lines are catalysts, and λ stands for no or inert species.

Here we propose a new and optimized variant of a chemical DL with a minimal error, minimal latency, minimal number of injections, and a low number of signals. We call it the parallel-accessible DL (PDL) because several nonconcurrent stages are executed in parallel as opposed to blocking sequential stages employed before. The PDL utilizes the same copy reaction $X_i^C \xrightarrow{S_1/S_2} X_{i+1} + X_{i+1}^C$, but also introduces a wait queue $X_i^{T_j} \xrightarrow{S_1/S_2} X_i^{T_{j+1}}$, where the

past concentrations are buffered for the required number of cycles.

Only two alternating signals S_1 (red) and S_2 (blue) drive the PDL's execution, regardless of its size. As shown in Figure 2(c), an injection of S_1 fires a simultaneous production of all the values X_1, X_2, X_3 , and other copy or move-inwait-queue reactions. The signal S_2 injected some time after S_1 triggers the remaining reactions that move values further to terminals X_i , consumed by an underlying system. The key idea here is that out of each pair of two neighboring species, there is always at least one with zero concentration since no S_1 (or S_2) signals drive the adjacent reactions. The concentration pathway from X to X_i contains 2i-1 reactions with i red and i-1 blue signals.

We set the rates of all PDL reactions to $k_{cat}=2$ and $K_m=0.075$. In order to fully proceed the S_1 phase in 25 steps, but at the same time prevent a signal overlap that could produce a transfer error, we set the rate of S_1 and S_2 decay to 0.6. Due to a negligible error and a constant latency, i.e., all past values are available immediately, the PDL could easily integrate with other chemical systems and provide them with a reliable and fast memory. Compared to our previous models, the new model, however, requires more species and reactions. The number grows quadratically for both the species $(\frac{(n+2)(n+1)}{2})$ and the reactions $(\frac{(n+1)n}{2})$. Table 1 summarizes the attributes of all our chemical DL models.

Table 1: Comparison of three different types of a chemical DL. _____

Attribute	Manual DL	Signal Backprop. DL	Parallel DL
Error Injections Latency Signals Species Reactions	$ \begin{array}{c c} O \\ O(n) \\ O(n) \\ O(n) \\ O(n) \\ O(n) \\ O(n) \end{array} $	$\begin{array}{c} \exp\\ O(1)\\ O(n)\\ O(n)\\ O(n)\\ O(n)\\ O(n) \end{array}$	$ \begin{array}{ c c } \hline & 0 \\ O(1) \\ O(1) \\ O(1) \\ O(n^2) \\ O(n^2) \\ \end{array} $

Perceptron Integration

In this section we will integrate an AASP (Section III) with a PDL (Section IV) to create a new memory-enabled AASP (AASP-DL) by feeding the species X_i produced by the PDL directly into an AASP as shown in Figure 3. As described before, that specific setup will allow us to perform temporal signal processing very effectively.

In order to operate with arbitrary memory we extend the AASP to more than two inputs. For each new cached input X_i we add five reactions: $X_i \xrightarrow{W_i} X_i Y + Y$ and $X_i + Y \to \lambda$ for the input-weight integration, and $W^{\oplus} \xrightarrow{X_i Y} W_i$, $W^{\ominus} \xrightarrow{X_i Y} W_i^{\ominus}$, and $W_i + W_i^{\ominus} \to \lambda$ for the positive and negative weight adaptation. The AASP of size n has therefore 4n+9 species and 5n+8 reactions. All new reactions use the same rate constants as the original X_1/X_2 reactions. Since the values X_i are produced rapidly, the

AASP's timing, such as the injection of the input signal S_{in} , are unchanged. Also, a single DL operational cycle of 50 time steps is compatible with 500 time steps required for the AASP's training cycle. To demonstrate the scalability, we model AASP-DLs of size 2 to 5.

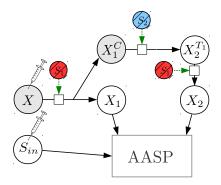


Figure 3: An AASP-DL schematic of size n = 2.

Experiments

In this section we describe the tasks, the performance metrics, the training setup, and the learning results of the AASP-DL.

Tasks

We have decided to use the following tasks to evaluate the performance of our new system. The selection and setting of temporal tasks reflect the fact that the expected AASP's output concentration must be between the minimal (zero) and the maximal output concentration, which is equal to the sum of all inputs provided $[S_{in}] + [X_1] + \ldots + [X_n]$.

1) LWMA2: Linear Weighted Moving Average (LWMA) is a time-series of a lagged averages, where each past element is weighted by an arbitrary value. The LWMA of order 2 is defined as

$$y_t = k_1 u_{t-1} + k_2 u_{t-2} + k_0,$$

where $k_1, k_2 \in (0.2, 0.8)$ are randomly drawn constants, $k_0 \in (0.1, 0.4)$ is a constant bias, and u_t is an i.i.d input stream generated uniformly from (0.2, 1). The task is to output y_t based on the past inputs u_{t-i} .

2) WMM2: The Weighted Moving Maximum (WMM) is a time-series of maximum lagged inputs. The WMM of order two is defined as

$$y_t = k \max(u_{t-1}, u_{t-2}) + k_0,$$

where similarly to LWMA2 the constants k and k_0 are randomly drawn from the interval (0.2,0.8) and (0.1,0.4) respectively, and u_t is an i.i.d input stream generated uniformly from (0.2,1). The task is to output y_t based on the past inputs u_{t-i} .

3,4) NARMA: The Nonlinear AutoRegressive Moving Average (NARMA) (Atiya and Parlos, 2000) is a discrete timeseries, where the current output y_t depends on both the previous inputs and outputs up to a given depth (order). The NARMA task of order n is defined as

$$y_t = \alpha y_{t-1} + \beta y_{t-1} \sum_{i=1}^n y_{t-i} + \gamma u_{t-n} u_{t-1} + \delta,$$

where $\alpha=0.3, \beta=0.05, \gamma=1.5, \delta=0.1$, and u_t is an i.i.d input stream generated uniformly from an interval (0,0.5). The task is to produce the output y_t based on the previous inputs u_{t-i} . NARMA is widely used as a benchmark task in the neural and reservoir computing literature (Jaeger, 2001; Maass et al., 2002) due to its nonlinearity and dependence on long-term memory.

We tackle two variants, NARMA2 and NARMA10, i.e., the second and tenth order of the problem. Since NARMA10 could be unstable, we bound the series by a non-linear tanh saturation function. Also, we scale the target stream y_t for NARMA2 by 2 to better fit the AASP's output range.

Performance Measures

We define performance by two standard error measures: symmetric absolute mean percentage error (SAMP) with values ranging from 0% to 100%

$$\mathrm{SAMP} = 100 \langle \frac{|y - \hat{y}|}{y + \hat{y}} \rangle$$

and the root normalized mean square error (RNMSE)

$$\text{RNMSE} = \sqrt{\frac{\langle (y - \hat{y})^2 \rangle}{\sigma_{\hat{y}}^2}},$$

where the square error is normalized by $\sigma_{\hat{y}}^2$, a variance of the target output \hat{y} . A RNMSE of 1 therefore corresponds to chance level.

Training Setup

The training of all AASP-DLs starts with a random setting of weight concentrations from (0.5, 1.5). During each training iteration we first inject the input X with a concentration corresponding to u_{t-1} for all tasks. Then we set the bias input S_{in} concentration to 0.5 for LWMA2 and WMM2, and 0.1 for NARMA2 and NARMA10. To trigger the DL operation we also provide the signal S_1 , which immediately produces both the current and the cached values X_i . To finish a PDL buffering procedure we inject another DL signal S_2 25 time steps later. Then again after 25 time steps we finally provide both the learning signal S_L and the target output \hat{Y} representing y_t to initiate the weight adaptation.

We run the AASP-DL against 10,000 training time series of length 800 for each task. We evaluate the RNMSE and SAMP performance over 10,000 runs for each training iteration, however, we report only the final training error.

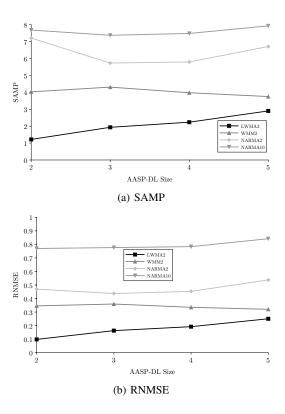


Figure 4: The relation between the final SAMP and RNMSE error and the AASP-DL size after 800 learning iterations. For each task 10,000 runs were performed.

Results

The AASP-DL reaches a relatively small error for all temporal tasks, which demonstrates that even a single chemical perceptron, due to its inherent nonlinearity, posses sufficient learning and computing capabilities. The error of the AASP-DL with optimal size settles to the range of (1.20%, 7.37%) for SAMP and (0.10, 0.77) for RNMSE as shown in Figure 4. Figures 5 and 6 show SAMP and RNMSE for all tasks over time.

The easiest function is a linear LWMA2 (SAMP of 1.20), with performance decreasing as memory of the past inputs grows. That is expected because LWMA2 depends only on the last two inputs u_{t-1} and u_{t-2} , so any additional information is essentially superfluous. Note that the AASP cannot fully eliminate the contribution or consumption of an extra input-weight branch, hence the input here basically acts as a noise. Figure 7 shows an example of the weight concentration traces and the filtered output and target output for LWMA2.

The WMM2 task's output is also fully prescribed by the last two inputs, however as opposed to LWMA2, the AASP-DL of size 5 performs best (SAMP of 3.75) with a marginal difference to the n=2 instance (SAMP of 4.02). Even

though the extra past inputs do not affect the target output y_t , the AASP might utilize them as a statistical variance source.

Because of its recurrent definition, the NARMA2 performance improves significantly for a longer DL, reaching an optimum for n=3 (SAMP of 5.73) and slightly worse for n=4 (SAMP of 5.81). Since the NARMA2 depends on the last two inputs u_{t-1} and u_{t-2} and recurrently on the last two outputs y_{t-1} and y_{t-2} , its output is fully determined by the last four inputs, which is inline with our results. NARMA10 is the most difficult task (SAMP of 7.37) due to the function dependence on long lags. The performance is fairly constant for $n\leq 4$.

Discussion and Conclusion

In this paper we demonstrated that the idea of chemical learning can be extended to temporal tasks by utilizing a memory of past concentrations provided through a chemical delay line. The AASP-DL successfully learns to produce a weighted moving average as well as a moving maximum. These operations could be applied to monitoring certain substances in a patient's blood, such as the insulin level, and perhaps also to control it by a conditional release of a specific amount of a required substance (cure). To demonstrate more complex nonlinear time dependencies, we also trained the AASP-DL for both the NARMA2 and NARMA10 benchmark tasks.

Our results show that memory improves learning for the recursive NARMA2 and nonlinear WMM2 tasks, but leads to lower performance for the simple LWMA2 task. Because the weights compete with each other and the AASP cannot fully eliminate the contribution or consumption of an extra input-weight branch, for a memory larger than the task inherently requires, performance decreases. Note that for the initial weights drawn uniformly from the interval (0.5-1.5) the AASP's ideal output region optimized by genetic algorithms (Banda and Teuscher, 2014) is around half of the maximal output concentration, which equals the total amount of input injected $[S_{in}] + [X_1] + [X_2]$. To move the input-output relation closer to that region we scaled the NARMA2 task by two.

Furthermore, since the AASP was optimized for the input range (0.2,1) with just two inputs, we can speculate that for a larger memory (and number of inputs), some part of the performance decrease is due to the fact that the output surface for the usual weight concentrations moves away from the responsive region and becomes either flat (saturated) or highly rugged, where even a small perturbation of a weight concentration changes the output significantly. Also, since an optimal weight learning region is nonlinear, we can expect that scaling the input and output by the AASP-DL size would not necessarily help. An optimal scaling of the AASP-DL's input and output concentrations is an important topic that is, however, beyond the scope of this paper.

To further improve the performance, the next step would

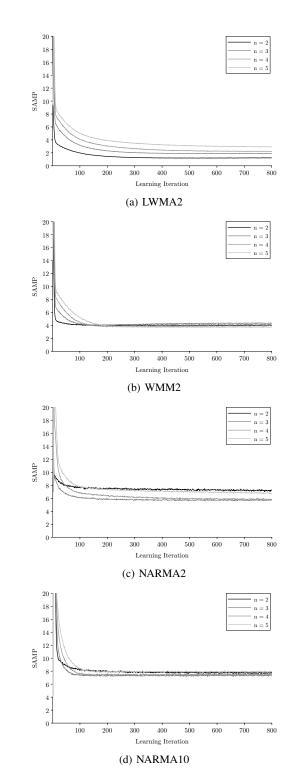


Figure 5: The SAMP error over time for all tasks. Average values over 10,000 runs.

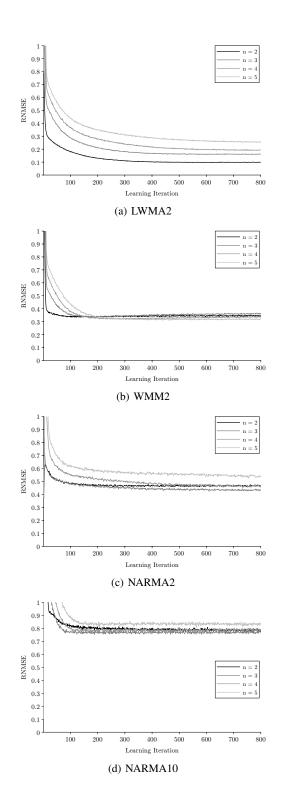


Figure 6: The RNMSE error over time for all tasks. Average values over 10,000 runs.

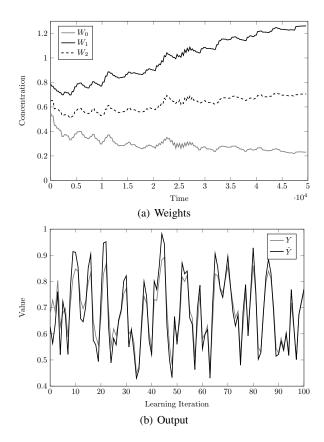


Figure 7: An example of the AASP-DL n=2 learning LWMA2, showing (a) the concentration traces of the weights and (b) the filtered output and the target output.

be to compose several chemical perceptrons with a delay line into larger (nested) multicompartment networks, where each compartment hosts a single chemical perceptron and where compartments communicate with each other through a channel-mediated exchange of molecular species.

Note that the NARMA task is generally tackled with a magnitude larger and more complicated machine learning approaches than our single memory-enabled chemical perceptron. A common approach is to employ recurrent neural networks (Atiya and Parlos, 2000) or advanced reservoir computing (Büsing et al., 2010), i.e., echo state networks (Jaeger, 2001) or liquid state machines (Maass et al., 2002), consisting of hundreds of nodes (neurons). For instance, minimal complexity echo state networks reported by Tino and Rodan (Rodan and Tino, 2011) needed 50 nodes to achieve a NMSE of 0.16, i.e., RNMSE of 0.4. The primary goal of using NARMA in this paper was to train our simple chemical learner to demonstrate its principal capabilities and to set a baseline.

We also plan to implement our model in an actual wet chemistry. More specifically our goal is to map each catalytic reaction to a deoxyribozyme-based substrate cleavage (Stojanovic and Stefanovic, 2003; Liu et al., 2009). In the core deoxyribozyme reaction $QF \xrightarrow{D} Q + F$, the downstream enzyme (catalytic DNA) D cleaves the oligonucleotide (short, single-stranded DNA) QF into two parts Q and F, where F is a chemical called fluorescein emitting fluorescence, which we can measure. By combining different cascading types of upstream and downstream enzymes and activation and deactivation relations, more complicated scenarios could be obtained. However, the mapping of three enzymes X_1Y, X_2Y , and $S_{in}Y$, as well as the DL operational signals S_1 and S_2 , which catalyze two (or more) reactions, would be non-trivial to implement in practice. Hence, we could expect that each of these catalysts would need to have several variants to serve different reactions. Alternatively we could obtain a wet chemical implementation of the AASP-DL by compiling mass-action-driven CRN to DNA-strand displacement reactions (Soloveichik et al., 2010; Zhang and Seelig, 2011). That would, however, produce more than 100 different strands, which would be complicated to produce accurately in the lab. Finally, note that for DNA-strand displacement, we would need to continually add new gate structures (fuel) to the solution to make the circuit reusable many times during training.

Acknowledgment

This material is based upon work supported by the National Science Foundation under grant no. 1028120.

References

- Atiya, A. F. and Parlos, A. G. (2000). New results on recurrent network training: unifying the algorithms and accelerating convergence. *Neural Networks, IEEE Transactions on*, 11(3):697–709.
- Banda, P. and Teuscher, C. (2014). Learning two-input linear and nonlinear analog functions with a simple chemical system (in press). In Ibarra, O. H. and Kari, L., editors, *Unconventional Computing and Natural Computing Conference*, volume 8553 of *Lecture Notes in Computer Science*, pages 14–26. Springer International Publishing Switzerland.
- Banda, P., Teuscher, C., and Lakin, M. R. (2013). Online learning in a chemical perceptron. *Artificial life*, 19(2):195–219.
- Banda, P., Teuscher, C., and Stefanovic, D. (2014). Training an asymmetric signal perceptron through reinforcement in an artificial chemistry. *Journal of The Royal Society Interface*, 11(93).
- Bray, D. (1995). Protein molecules as computational elements in living cells. *Nature*, 376(6538):307–312.
- Büsing, L., Schrauwen, B., and Legenstein, R. (2010). Connectivity, Dynamics, and Memory in Reservoir Computing with Binary and Analog Neurons. *Neural Computation*, 22(5):1272–1311.
- Copeland, R. A. (2002). *Enzymes: A practical introduction to structure, mechanism, and data analysis.* John Wiley & Sons, Inc., New York, New York, second edition.

- Dittrich, P., Ziegler, J., and Banzhaf, W. (2001). Artificial chemistries a review. *Artificial Life*, 7(3):225–275.
- Espenson, J. (1995). Chemical kinetics and reaction mechanisms. McGraw-Hill, Singapore.
- Haykin, S. (2009). Neural networks and learning machines. Pearson, New Jersey, third edition.
- Jaeger, H. (2001). The echo state approach to analysing and training recurrent neural networks-with an erratum note. Bonn, Germany: German National Research Center for Information Technology GMD Technical Report, 148:34.
- Kim, J., Hopfield, J. J., and Winfree, E. (2004). Neural network computation by in vitro transcriptional circuits. In Saul, L. K., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems*, volume 17, pages 681–688. MIT Press.
- LaVan, D. A., McGuire, T., and Langer, R. (2003). Small-scale systems for in vivo drug delivery. *Nature biotechnology*, 21(10):1184–91.
- Liu, J., Cao, Z., and Lu, Y. (2009). Functional nucleic acid sensors. *Chemical Reviews*, 109(5):1948–1998. PMID: 19301873.
- Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560.
- Mills, A. P., Yurke, B., and Platzman, P. M. (1999). Article for analog vector algebra computation. *Biosystems*, 52(1-3):175– 180.
- Moles, J., Banda, P., and Teuscher, C. (2014). Delay line as a chemical reaction network (under review). *Parallel Processing Letters*, http://arxiv.org/abs/1404.1152.
- Qian, L., Winfree, E., and Bruck, J. (2011). Neural network computation with DNA strand displacement cascades. *Nature*, 475(7356):368–372.
- Rodan, A. and Tino, P. (2011). Minimum complexity echo state network. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 22(1):131–44.
- Rojas, R. (1996). Neural networks: A systematic introduction. Springer-Verlag, Berlin.
- Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organisation in the brain. *Psychological Review*, 65:368–408.
- Soloveichik, D., Seelig, G., and Winfree, E. (2010). DNA as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences of the United States of America*, 107(12):5393–5398.
- Stojanovic, M. N. and Stefanovic, D. (2003). A deoxyribozymebased molecular automaton. *Nature Biotechnology*, 21(9):1069–1074.
- Zhang, D. Y. and Seelig, G. (2011). Dynamic DNA nanotechnology using strand-displacement reactions. *Nature chemistry*, 3(2):103–113.

Chemlambda, universality and self-multiplication

Marius Buliga ¹ and Louis H. Kauffman ²

¹ Institute of Matematics of the Romanian Academy P.O. BOX 1-764, RO 014700, Bucharest, Romania

Marius.Buliga@gmail.com

² Department of Mathematics, University of Illinois at Chicago 851 South Morgan Street, Chicago, Illinois, 60607-7045

kauffman@uic.edu

Abstract

We present chemlambda (or the chemical *concrete* machine), an artificial chemistry with the following properties: (a) is Turing complete, (b) has a model of decentralized, distributed computing associated to it, (c) works at the level of individual (artificial) molecules, subject of reversible, but otherwise deterministic interactions with a small number of enzymes, (d) encodes information in the geometrical structure of the molecules and not in their numbers, (e) all interactions are purely local in space and time. This is part of a larger project to create computing, artificial chemistry and artificial life in a distributed context, using topological and graphical languages.

Introduction

In this note we want to briefly present chemlambda (or the chemical *concrete* machine), an artificial chemistry with the following properties: (a) is Turing complete, (b) has a model of decentralized, distributed computing associated to it (Buliga and Kauffman, 2013), (c) works at the level of individual (artificial) molecules, subject of reversible, but otherwise deterministic interactions with a small number of enzymes, (d) encodes information in the geometrical structure of the molecules and not in their numbers, (e) all interactions are purely local in space and time.

In some respects chemlambda is closed to the fraglets (Tschudin, 2003) and metabolic approaches (Tschudin and Yamamoto, 2004) research line. In others, it resembles to the CHAM ("chemical abstract machine") (Berry and Boudol, 1992), which uses a chemical metaphor for modeling asynchronous concurrent computations (in particular a concurrent lambda calculus). Algorithmic Chemistry (Fontana and Buss, 1996) (Fontana and Buss, 1994a) (Fontana and Buss, 1994b) is another classical line of inspiration. Because it concentrates at the level of individual molecules, it departs however from the programming model of computation introduced in (Banâtre and Le Métayer, 1986) (Banâtre et al., 1988).

Chemlambda appeared as an artificial chemistry version of a graph rewrite system, called graphic lambda calculus (GLC) (Buliga, 2013b) (web tutorial). In GLC programs are

certain trivalent graphs, and execution of programs means the application of graph rewrites, called "moves", on the respective graph.

In the GLC formalism there is one global move (GLOBAL FAN-OUT), all the other moves are local (i.e. they involve a fixed, small number of nodes).

Chemlambda was introduced in order to eliminate this unpleasant GLOBAL FAN-OUT. Chemlambda uses only local moves (Buliga, 2013a) (web tutorial). The moves of chemlambda act on trivalent graphs called "molecules" at certain "reaction sites", like chemical reactions involving molecules and enzymes (here enzyme=move).

Later on, a distributed, decentralized model of computation appeared, called distributed GLC (Buliga and Kauffman, 2013), which is based on chemlambda and GLC, also using the Actor Model by Hewitt (Hewitt, 2010) (Agha, 1986). Distributed GLC models asynchronous, distributed, decentralized computations. The Actor Model ingredient is a replacement of proximity relations between (individual) interacting molecules. Indeed, real chemical interactions happen only between molecules which are close one to another. But in the chemlambda formalism there is no space where these artificial molecules float, they are just certain trivalent graphs, not embedded in any way in a space. Computation with chemlambda molecules is seen as asynchronous, purely local and decentralized application of graph rewrites (i.e. moves, or interaction with enzymes). Proximity relations are then replaced by interactions between actors, each actor being in charge of a molecule, and having a very limited repertoire of behaviours. (In turn, each behaviour uses one of the graph rewrites available, either applied between two interacting actors, or internally, as it is the case of the sequences of moves which effect a selfmultiplication replacing the GLOBAL FAN-OUT of GLC).

The key merit of this model is a graphical reformulation of the well-known lambda calculus, central to logic and to the design of recursion in computer languages. By reformulating the lambda calculus in terms of graphs, the operations for the calculus become essentially local operations of graphical replacement. The graphs themselves contain

all the data that is usually formulated in terms of algebra. This means that the global structure of the graph contains all the information that is usually cut up into bits of algebra. The graph becomes a whole system that instantiates the computational power of the calculus. This instantiation is the key reason why this model can propose significant designs in distributed computing. The graph as a whole can exist in a widely distributed fashion, while the interactions that constitute its computations are controlled by local nodal exchanges between actors.

Furthermore, this property of redesigning the relationship of the local and the global is not restricted just to lambda calculus networks. Chemlambda may be used as well with other models than distributed GLC, like L-systems or chemical reaction networks. There are relationships of the same kind that link this research with topology (knots and lambda calculus (Kauffman, 1994b), knot automata (Kauffman, 1994a)), or with topological quantum computing (Chen et al., 2007), (Kauffman and Lomonaco Jr., 2006).

A quick review of lambda calculus

In this section we give a very quick review of the formalism and ideas of lambda calculus. First of all the notation

$$F = \lambda x y. f(x, y)$$

indicates a function f(x,y) of two variables, defined in some domain and a stipulation (the part after λ and before the function) of the order of application of the operator F to these variables. Thus we can write

$$(Fx)y = f(x,y).$$

For example, If

$$F = \lambda x y. y(yx),$$

then

$$(Fa)b = b(ba).$$

Later we will make a notation for the operation of evaluating such an operator, but for now we just consider the non-associative algebra structure of such operators. We can work in reverse as well. Suppose I say that G is an operator defined by the equation

$$((Gx)y)z = (yx)(yz).$$

Then we have in the λ notation,

$$G = \lambda xyz.(yx)(yz).$$

For this analysis, let us suppose that the algebra generated by the variables x, y, z, \cdots is a universal non-associative algebra. This means that the binary operation is non-associative and there are no further relations instantiated. However, if we define M by Mx = xx and regard M as an element of an extension of the original algebra by giving

it the status of $M=\lambda x.xx$, then M satisfies the special relation that defines it and furthermore we would like to be able to say that the definition of the action of M applies even if we apply M to itself. In that case we would have MM=M as a consequence of the definition $M=\lambda x.xx$. Thus we can start with a universal non-associative algebra and then add new elements that satisfy special relations. We can in this freely made situation allow the new elements to act (compose) upon themselves.

Here is a useful example. Let F be a given operator. It can be one of the original variables, or it can be a defined operator such as we have discussed above. The we define G as

$$Gx = F(xx)$$
.

That is, we define

$$G = \lambda x.F(xx).$$

Now we note that

$$GG = F(GG)$$
.

Thus any F in our algebra has a fixed point that is another element of the algebra. This is the Fixed Point Theorem of Church and Curry. Along with this fixed point theorem comes some caution in the use and construction of such lambda calculi. For suppose we had been dealing with a logical calculus and $F=\sim$, the negation operator. Then in our initial calculus we may have assumed that negation does not have a fixed point, as in classical logic. But we have seen that if

$$G = \lambda x. \sim (xx),$$

then

$$GG = \sim (GG)$$
.

Thus the extended algebra can not be expected to continue to obey classical logical rules. If it is desired to continue to obey such rules then one must put some controls on the lambda calculus. Also, if one has a fixed point as in

$$GG = F(GG)$$
,

then there is the possibility of an infinite recursion of the form

$$GG = F(GG) = F(F(GG)) = F(F(F(GG))) =$$

$$= F(F(F(F(GG)))) = F(F(F(F(F(GG))))) = \cdots$$

It is good to have a formalism for recursion, but the language needs to include controls for that so that a computation does not run without stopping. One way to handle such control is to replace equality of evaluation by an evaluation or reduction step. Then one would have

$$(\lambda x. H(x))a \longrightarrow H(a)$$

where the arrow refers to a reduction step that can be performed. In this case, the step is called *beta - reduction*. In the rest of this paper, we show how to adapt such controlled lambda calculi to operations on graphs where steps of replacement from one graph to another correspond to operations like beta-reduction. The graphs, once they are formulated, have the advantage that the details of labeling in the algebra have disappeared into graphical connections and so certain complexities of lambda calculi are handled automatically. We envisage such graphical systems and their evolutions under computational steps such as beta-reduction as a new and powerful formulation of computation and information processing.

The Chemlambda formalism

Chemlambda is a graph rewriting system. It consists in a family of graphs, called "molecules" and a list of graph rewrites, called "moves". Every move is local, in the sense that there is an a priori upper bound on the number of nodes and arrows which are modified during the move.

A *molecule* is a locally planar graph made by a finite number of pieces (arrows, loops, nodes described in Fig. 1). We may admit also a set of nodes with unspecified valences, called "other molecules". These are the equivalent of "cores" from (Buliga and Kauffman, 2013) section 3, paragraph 5. Interaction with cores, i.e. they can be used as interfaces with external constructs.

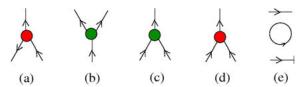


Figure 1: Basic pieces of chemlambda molecules: (a) lambda abstraction node, (b) fanout node, (c) application node, (d) fanin node, (e) arrow, loop and termination node

Each chemlambda *move* is to be interpreted as the interaction of a chemlambda molecule with an enzyme (which has the same name as the move), at a certain reaction site (the place where the move is applied). The moves are reversible.

The list of moves is the following:

- the beta move, Fig. 2 up, is the graphic version of beta reduction from lambda calculus. It is a local graph rewrite version of the Wadsworth (Wadsworth, 1971) or Lamping (Lamping, 1990) beta reduction move, in the sense that it can be applied whenever is possible, independently of

the fact that the graph, or molecule, represents a lambda calculus term.

the FAN-IN move, Fig. 2 down, is a dual of the beta move, involving a fanin and fanout node. It is as important as the beta move, being involved into self-multiplication of molecules.

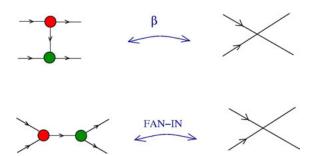


Figure 2: (up) the beta move, (down) the FAN-IN move

 the DIST moves, Fig. 3, provide the mechanism of selfmultiplication of molecules, together with the FAN-IN move

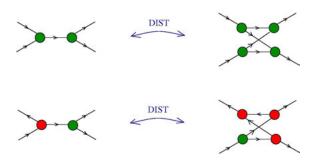


Figure 3: the DIST moves

- the co-associativity and co-commutativity moves, Fig. 4, are a very weak description of the fanout node as a fanout, in the sense that if we think about the fanout node as being a gate with one input and two outputs which are identical with the input, then graphically such a gate would satisfy the CO-ASSOC and CO-COMM moves. However, the fanout node is not a gate in this formalism, because there is nothing which propagates through the arrows of chemlambda molecules.
- the local pruning moves, Fig. 5, are useful in both senses, either as moves which destroy the "dead" arrows and nodes, or as moves which enrich the molecule by creating new arrows or nodes.

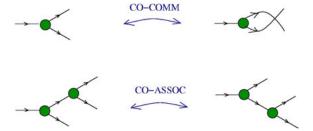


Figure 4: the co-associativity and co-commutativity moves

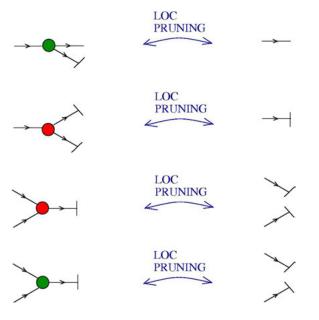


Figure 5: the local pruning moves

Chemlambda and lambda calculus

Lambda calculus combinators can be encoded as chemlambda molecules. In (Buliga, 2013a) Theorem 4.2 is given the encoding of the BCKW system of combinators from Fig. 6. The proof of the theorem has two parts: (a) the reduction relations of the BCKW system can be done in chemlambda, (b) the B,C,K,W combinator molecules *can reproduce, or self-multiply*. The conclusion of the theorem is that *chemlambda is Turing universal*.

We think it is interesting to explain in detail what this selfmultiplication means in the chemlambda formalism.

Remark, after inspection of the Fig. 6, that every combinator molecules has one arrow which points outwards from the molecule, let's call this arrow the "exit arrow". Recall that we have a fanout node among the basic pieces of chemlambda molecules. In order to prove the Turing universality, we need to be able to transform, by a sequence of chemlambda moves, one combinator molecule with the exit arrow connected to the in arrow of a fanout node, into two copies of the combinator molecule. We call this self-multiplication. (In the GLC formalism this self-multiplication is done via

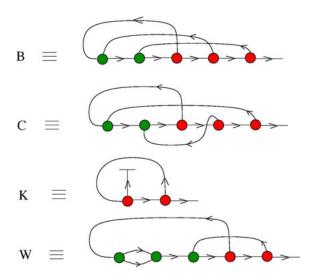


Figure 6: B,C,K,W combinators encoded in chemlambda

the move GLOBAL FAN-OUT, but chemlambda has only local moves.)

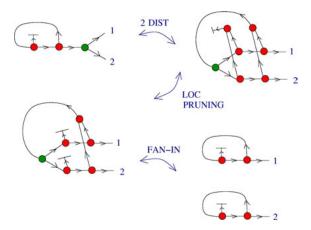


Figure 7: Self-reproduction of the K combinator molecule

As an example, in the Fig. 7 we see how the K combinator molecule self-reproduces, after a string of chemlambda moves.

Propagators, distributors, multipliers and guns

The self-multiplication of combinator molecules is done by a sequence of local moves of chemlambda. The sequence of moves depends on the combinator molecule. We have seen that self-multiplication is an important ingredient for proving Turing universality of chemlambda.

Many chemlambda molecules don't encode combinators or lambda calculus terms, moreover, moves like DIST or FAN-IN don't have a clear meaning as seen from the point of view of lambda calculus. The phenomenon of self-multiplication is not restricted to combinator molecules.

Let us then explore a bit the chemlambda formalism from the point of view of phenomena like self-multiplication, without caring about lambda calculus.

In the Fig. 8 are defined multipliers, propagators and distributor molecules. A chemlambda molecule with an exit arrow $A \rightarrow$ is a multiplier if there is a sequence of chemlambda moves, denoted by $MULT_A$, which produces the self-multiplication of the molecule. For example, any combinator molecule is a multiplier, but there are other multipliers as well.

A chemlambda molecule $\rightarrow A \rightarrow$ with distinguished in and out arrows is a propagator if there is a sequence of chemlambda moves, denoted by $PROP_A$, with the effect described in the second row of Fig. 8. The molecule is called a propagator because it looks like it propagates through the fanout nodes.

There are two kinds of distributor molecules, described in the 3rd and 4th rows of Fig. 8. Compare with the DIST moves from Fig.3, which can be interpreted by saying that the application node is a distributor of the first kind and the lambda abstraction node is a distributor of the second kind.

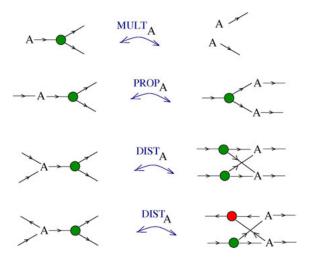


Figure 8: Definition of self-multipliers, propagators, distributors

Starting from the mentioned multipliers and distributors, we can make many other interesting molecules. For example, we can make a propagator from a multiplier A and a distributor of the first kind B, as described in Fig. 9.

In Fig 10 is described a multiplier made from a distributor of the second kind.

We can as well make guns, which shoot an endless string of molecules, like in the Fig. 11. On the first row is described a gun made from a propagator molecule and a fanout node. On the second row is described a gun made from a distributor of the first kind and a fanout node.

See also (Buliga, 2013a) Section 3 for other examples of interesting chemlambda molecules, like zippers, sets and pairs.

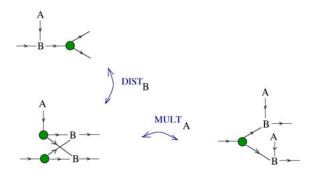


Figure 9: Propagator made from a multiplier and a distributor of the first kind

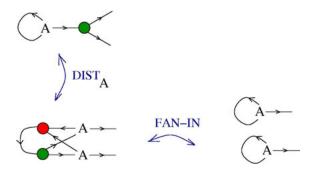


Figure 10: Multiplier made from a distributor of the second kind

All these constructions show that we can use chemlambda for building all sorts of synchronous or asynchronous automata (which are not living on a predefined lattice, instead they grow their own lattice). Also, we proved the potential of chemlambda to evolve complex molecules from simple ones.

The Y combinator and self-multiplication

In this section we come back to lambda calculus, in order to explain the behaviour of the Y combinator molecule. From the previous sections we learned that self-multiplication is a basic ingredient for encoding the BCKW system of combinators in chemlambda. The moves applied to a combinator molecule represent the reduction of the combinator. Self-multiplication is needed in order to produce copies of a part of the combinator molecule, with the purpose of further reducing one of the copies, while having at our disposal the other copy for further needs.

Seen like this, it seems that self-multiplication is also a basic ingredient for recursion. In lambda calculus there is the iconic Y combinator which represents the essence of recursion. In the following we shall see that, however, self-multiplication is not directly needed in the reduction of the Y combinator.

The Y combinator has the expression

$$Y = \lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$$

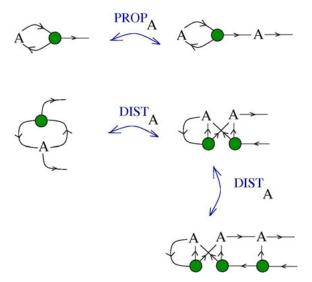


Figure 11: Examples of guns

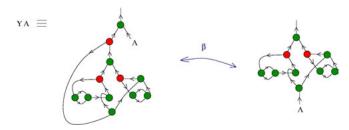


Figure 12: the YA combinator molecule and a first beta move

and it has the following property: for any lambda term A the expression YA reduces to A(YA). In particular, if A is another combinator, then YA is a fixed-point combinator for A.

In lambda calculus the string of reductions is the following sequence of beta moves:

$$YA \to (\lambda x. A(xx))(\lambda x. A(xx)) \to$$
$$\to A((\lambda x. A(xx))(\lambda x. A(xx))) = A(YA)$$

We see that the during the reduction process we needed a multiplication of the combinator A.

Let us pass to the chemlambda encoding of the Y combinator. With A another combinator molecule, the combinator molecule which encodes YA is the one from the left hand side of the Fig. 12.

After the application of a beta move, it transforms into the molecule from the right hand side of Fig. 12. Continuing from the Fig. 12, there is a second beta move which can be applied, as in Fig. 13.

There are two DIST moves, one of the first kind, the other of the second kind, which are applied, as in Fig. 14.

Let's see how we can reduce further this molecule, until we obtain one which corresponds to A(YA). We shall use

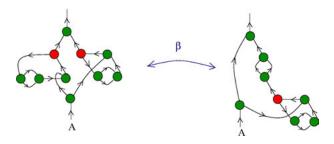


Figure 13: second beta move applied to the YA molecule

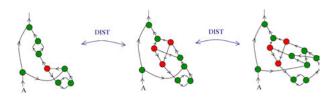


Figure 14: next step of reduction, two DIST moves

the fact that a certain molecule, called *the bit* is a propagator, as proved in Fig. 15. The bit molecule corresponds to the expression (xx) which appears repeatedly in the Y combinator.

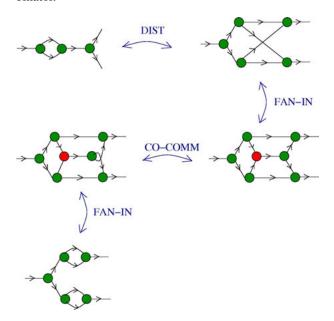


Figure 15: the bit is a propagator

We continue from the Fig. 14 and we apply the PROP move of the bit and then a FAN-IN move, as in the Fig. 16.

The last molecule corresponds to A(YA), if we interpret the fanout nodes as real fan-out gates.

Surprisingly, during the reduction there was no need to use the fact that the combinator molecule A is a multiplier! This shows that the Y combinator molecule can be used as a fixed point combinator with any other chemlambda

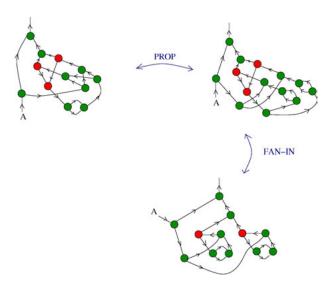


Figure 16: last two moves of the reduction of YA to A(YA)

molecule. That is because the Y combinator molecule is a gun which shoots fanout nodes, Fig. 17.

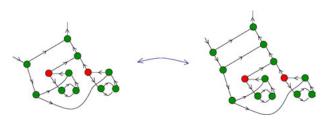


Figure 17: the Y molecule is a gun

A topological version of chemlambda

In (Buliga and Kauffman, 2013) Section 5 is proposed a topological version of GLC, called TGLC. We can do the same with chemlambda. The idea is that we may imagine formalisms which are equivalent with GLC and chemlambda, even if visually they seem different.

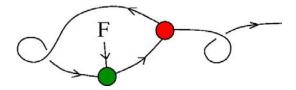


Figure 18: Topological Fixed Point Combinator

For a topological version of chemlambda we may use some of the basic nodes of chemlambda together with knot diagrams crossings. In Fig. 19 we give two possible translations of crossings into chemlambda: (a) as a pair of a fanout and application node, corresponding to the proposal made in (Buliga and Kauffman, 2013) Section 5, or (b) as a pair

of a lambda abstraction node and an application node, as proposed in (Buliga, 2013b) Section 6. A crossing is a 4 valent vertex. *Virtual crossings*, i.e. encircled crossings of graphical lines, may be used for making our graphs globally planars instead of only locally planar, as previously.

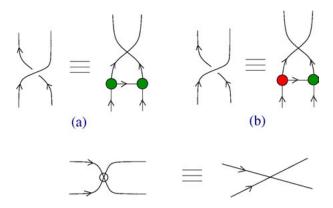


Figure 19: first row, two possible translations from crossings to chemlambda, second row a virtual crossing

If we stick to the choice (a) then we obtain a topological version of chemlambda, that has the form of knot diagrams equipped with extra lambda nodes and multiplication nodes.

In Fig. 18 we illustrate the basic fixed point combinator

$$G = \lambda x.F(xx)\lambda x.F(xx)$$

In this knot diagrammatic convention, the two self-multiplications that occur at two levels in this expression are instantiated by the two curls in the graph.

Similarly, in Fig. 20 we illustrate a topological expression for the *Y*-combinator.

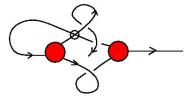


Figure 20: Topological Y - Combinator

Note how the structure of this combinator takes on the hybrid nature of tangle diagram infused with curls and lambda nodes. It is natural to use virtual crossings in graph theory and in fact there is an extension of knot theory that allows exactly such virtual crossings in the knot diagrams.

In Fig. 21 we see that, via a CO-COMM move, a curl is a bit, the molecule which appears in Fig 15.

The fact that alpha reduction is not needed in chemlambda due to the absence of variables and the presence of direct connections that effect interactions is part of a link of this formalism with the formalisms at the knot theoretic and



Figure 21: a curl is a bit

topological level. One difference between knot theoretic considerations and lambda calculus considerations is in the fact that we do not usually think of a knot diagram as a computing element that undergoes moves and reductions for the sake of a computation. But this is not always so. For example, the skein algorithms such as the bracket polynomial algorithm can be regarded as a reduction process that produces two new graphs from each crossing in the knot diagram. This is similar to allowing free beta reduction in the lambda calculus graphs. What must be done however in the knot theoretic case is to collect up all the end calculation results and add them together. This is what is meant by a formula like

$$\langle K \rangle = \Sigma_S \langle K | S \rangle.$$

(See (Kauffman, 1991).) Each S is a pattern of reductions leading to a specific algebraic value $\langle K|S\rangle$. The topological invariance occurs at the level of the sum of all of these contributions.

An analogous situation could occur in a stochastic version of chemlambda, where one would need the average over all the results of the many branching graph rewrites.

References

- Agha, G. (1986). Actors: A model of concurrent computation in distributed systems. Doctoral Dissertation. MIT Press, (html).
- Banâtre, J.-P., Coutant, A., and Le Métayer, D. (1988). A parallel machine for multiset transformation and its programming style. *Future General Computer Systems*, 4:133–144.
- Banâtre, J.-P. and Le Métayer, D. (1986). A new computational model and its discipline of programming. Technical Report INRIA Report 566.
- Berry, G. and Boudol, G. (1992). The chemical abstract machine. *Theoretical Computer Science*, 96(1):217–248.
- Buliga, M. (2013a). Chemical concrete machine. arXiv:1309.6914.
- Buliga, M. (2013b). Graphic lambda calculus. *Complex Systems*, 4:311–360.
- Buliga, M. and Kauffman, L. (2013). Gle actors, artificial chemical connectomes, topological issues and knots. arXiv:1312.4333.
- Chen, G., Kauffman, L., and Lomonaco, S. (2007). *Mathematics in Quantum Computation and Quantum Technology*. Chapman & Hall/CRC.

- Fontana, W. and Buss, L. (1994a). "the arrival of the fittest": Toward a theory of biological organization. *Bull. Math. Biol.*, 56:1–64.
- Fontana, W. and Buss, L. (1994b). What would be conserved if the tape were played twice? *Proc. Natl. Acad. Sci. USA*, 91:757–761.
- Fontana, W. and Buss, L. (1996). The barrier of objects: From dynamical systems to bounded organizations. In J.Casti and J., Karlqvist, A., editors, *Boundaries and Barriers*, pages 56–116. Addison-Wesley.
- Hewitt, C. (2010). Actor model of computation. arXiv:1008.1459.
- Kauffman, L. (1991). Knots and Physics. World Scientific Pub.
- Kauffman, L. (1994a). Knot automata. In Proceedings of The Twenty-Fourth International Symposium on Multiple-Valued Logic, Boston, Massachusetts, pages 328–333.
- Kauffman, L. (1994b). Knot logic. In Kauffman, L., editor, *Knots and Applications*, pages 1–110. World Scientific Pub.
- Kauffman, L. and Lomonaco Jr., S. (2006). q deformed spin networks, knot polynomials and anyonic topological quantum computation. quant-ph/0606114.
- Lamping, J. (1990). An algorithm for optimal lambda calculus reduction. In POPL '90 Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, pages 16–30.
- Tschudin, C. (2003). Fraglets a metabolistic execution model for communication protocols. In *Proc. 2nd Annual Symposium on Autonomous Intelligent Networks and Systems (AINS), Menlo Park, USA.*
- Tschudin, C. and Yamamoto, L. (2004). A metabolic approach to protocol resilience. In *Proc. 1st Workshop on Autonomic Communication (WAC 2004), Berlin, Germany*, pages 191–206. Springer LNCS volume 3457.
- Wadsworth, C. (1971). Semantics and pragmatics of the lambda calculus. DPhil thesis, Oxford.

Self-Organising Autocatalysis

Nathaniel Virgo^{1,3}, Simon McGregor^{2,4} and Takashi Ikegami^{1,5}

¹Ikegami Laboratory, University of Tokyo, Japan ²University of Sussex, UK ³nathanielvirgo@gmail.com ⁴s.mcgregor@sussex.ac.uk ⁵ikeg@sacral.c.u-tokyo.ac.jp

Abstract

Life on Earth must originally have arisen from abiotic chemistry. Since the details of this chemistry are unknown, we wish to understand, in general, which types of chemistry can lead to complex, life-like behaviour. Our recent work has shown that the inclusion of thermodynamic principles in simple artificial chemistry models can result in the selforganisation of autocatalytic cycles. In this paper we present some new insights into why this happens. Our model is given a more mathematical treatment, allowing us to better understand the assumptions that lead to this phenomenon. The simplest type of autocatalytic cycle results in exponential growth. Through dynamical simulation we demonstrate that when these simple first-order cycles are prevented from forming, the system achieves super-exponential growth through more complex, higher-order autocatalytic cycles. This leads to non-linear phenomena such as oscillations and bistability, the latter of which is of particular interest regarding the origins of life.

Introduction

How can abiotic chemistry give rise to phenomena such as replication, metabolism and evolution? This question is at the core of the origins of life as a field of inquiry, and there are many approaches to it. One such methodology is *artificial chemistry*, the study of more-or-less abstract artificial chemical reaction networks and their emergent dynamical properties. This field can be traced back to the work of Fontana and Buss (1994) and Kauffman (1986). More recently it has been used to show that evolution can occur without the need for template replication (Vasas et al., 2012), and to investigate the mechanism behind HCN polymerisation, a prebiotically-relevant reaction that is too complex to model with traditional approaches (Andersen et al., 2013).

In these models, thermodynamic principles play at best a minor role. However, the nature of living organisms as far-from-equilibrium structures is fundamental to our understanding of biology at the chemical level. Our work aims to show that new insights can be gained by putting thermodynamics at the heart of artificial chemistry approaches.

Last year we presented a simple yet thermodynamically reasonable artificial chemistry (Virgo and Ikegami, 2013).

When held out of equilibrium, this chemistry formed moderately large and complex autocatalytic cycles. (That is, moderately large sets of species that are collectively capable of exponential growth.) Unlike many previous studies, this chemistry did not include any *a priori* notion of catalysis. Instead, catalytic effects emerged as dynamical properties of the reaction network. The system was allowed to dynamically "choose" the direction of its reactions according to thermodynamic principles, and it was this property that allowed it to self-organise into autocatalytic cycles.

This phenomenon of self-organising autocatalysis happens under some circumstances but not others. In order to fully understand the result we must investigate which classes of reaction networks will lead to autocatalysis and which will not. Moreover, the resulting cycles can be more or less complex, and we wish to understand what factors affect this. A comprehensive answer to this question would be a great help in understanding the origins of life. If we know which types of chemical system are likely to lead to complex, metabolism-like reactions then it will give us clues about what to look for in identifying the types of abiotic chemistry from which life eventually arose.

In this paper we continue this project. We present some new insights into why autocatalysis occurs in far-from-equilibrium systems, and into what is needed for this to happen. As with all non-equilibrium phenomena, the autocatalysis in our systems is caused by the tendency of energy (and other conserved currents) to "flow downhill" from more to less constrained states, i.e. from low to high entropy. Informally, it seems that nature likes to take the "path of least resistance" in order to achieve this. If the easiest path is a simple, direct chemical reaction then no complex behaviour will be observed. Without a "direct" pathway from initial conditions to equilibrium, autocatalytic pathways tend to be observed. Blocking the simplest type of autocatalytic cycle results in the formation of more complex cycles, giving rise to non-linear phenomena such as bistability and oscillations.

Autocatalytic cycles can only emerge if they are possible within the reaction network. However, if the direction in which reactions occur is not fixed *a priori* then even the

simplest reaction schemes contain many autocatalytic subnetworks.

The results we present are very robust. With one minor exception (the oscillatory behaviour shown in Figure 3) they require no parameter tuning at all. Moreover, our mathematical arguments are applicable to a much broader class of chemical systems than the specific models we present. Thus, although our model is relatively simple and abstract, we expect that in future work similar results will be found in much more realistic scenarios.

Thermodynamics in Artificial Chemistry

A glance at a chemistry text book will reveal a large number of formulae that appear to be largely empirical in nature, and this can give the impression that chemical thermodynamics is something very contingent upon the details of molecular interactions. Perhaps because of this, there is a tendency in artificial chemistry to "abstract away" thermodynamics, choosing a set of reactions arbitrarily, without regard to energetic considerations. This may be justified by saying that some of the reactions are driven by an energy source that is external to the system and is not explicitly modelled. This methodology dates back to the origins of artificial chemistry in the work of Fontana and Buss (1994).

Such an abstraction is of course perfectly valid, but we hope to show that important insights can be gained if we choose not to make it. The dynamics of a system where there are only a few energy sources and the majority of reactions must "flow downhill" are quite different from those in which the reactions proceed in arbitrarily-chosen directions. Moreover, the flow of free energy through the network can be tracked, giving us insights into its dynamics that would be missing if the energetics were not modelled.

Although chemical thermodynamics might appear to be a rather empirical set of results, its core principle (the second law) is an extremely fundamental property of physical systems. Essentially, it boils down to a requirement that for a system with no external supply of energy, there must exist a global Lyapunov function, which may be called the entropy or the free energy, depending on whether the system is connected to a heat bath. The existence of such a function is derived from statistical mechanics, and is a consequence of the time-reversibility of physics on the microscopic level. The details of this formalism are beyond the scope of this paper and may be found, for example, in Kondepudi and Prigogine (1998). We mention it in order to emphasise that our results are less contingent upon the details of molecular interactions than they might at first appear.

The key feature of a thermodynamically reasonable artificial chemistry is that the reaction network doesn't specify the *direction* of the reactions. If the network includes a reaction, such as $A+B\to C$, it must also include the reverse reaction, $C\to A+B$. In general, these two reactions may proceed at different rates. We refer to the difference in their

rates as the *net rate* of the bidirectional reaction, $A+B \rightleftharpoons C$.

The dynamics must be such that, in the absence of any energy sources, the system will eventually reach a state of *detailed balance* in which the net rate of every reaction is zero. That is, every forward reaction must be balanced, pairwise, by its corresponding reverse reaction. (This point is of fundamental importance. In a general artificial chemistry a dynamical equilibrium can be formed by a cycle, such as $A \to B$; $B \to C$; $C \to A$. In a thermodynamic system with no driven reactions this is a physical impossibility.) Every net reaction must always proceed in the direction that takes the system closer to this equilibrium state. Thus, no reaction can occur at a non-zero net rate unless the system is either in a transient state or is being driven by an external power source.

In our model we use a simple implementation of these dynamics, known as *mass action kinetics*. Essentially this corresponds to a model of a well-mixed reactor in which molecules collide with each other at random. When two molecules collide they react with a probability determined by the "rate constant" of the reaction. Thermodynamics puts constraints on the values of the rate constants; these constraints are satisfied by the dynamics described below. (See Kondepudi and Prigogine (1998) for details of the formalism, and Virgo and Ikegami (2013) for its application to models of the type described below.)

The A-polymerisation model

When implemented in a thermodynamically plausible way, even very simple reaction schemes can give rise to interesting behaviour. Our model consists of *unary strings* A, AA, AAA etc., which we write as A_1, A_2, A_3, \ldots In principle these are countably infinite, but we impose a limit for computational reasons. These species can be thought of as (non-oriented) polymers, with A_1 (or simply A) representing a monomer, and A_i a polymer (or oligomer) of length i. All reactions are of the form

$$A_i + A_j \Longrightarrow A_{i+j}. \tag{1}$$

That is, two oligomers may join together into a longer one, or an oligomer may split into two smaller ones. Reactions of this form conserve the total number of A monomers. We take the forward and reverse reaction rates to be constants $k_{\rm s}$ and $k_{\rm d}$, (for "synthesis" and "decomposition") with the same value for every reaction.

This corresponds to an assumption that a constant amount of energy $E_{\rm AA}$ is involved in every A-A bond (Virgo and Ikegami, 2013). Briefly, the free energy of formation of species A_i is given by $G_{A_i}^{\circ} = (i-1)E_{\rm AA}$ (plus other terms that cancel), and the equilibrium constant for reaction $A_i + A_j \rightleftharpoons A_{i+j}$ is given by $\frac{k_s}{k_d} = \exp \frac{G_{i+j}^{\circ} - G_i^{\circ} - G_j^{\circ}}{k_B T} = \exp \frac{E_{\rm AA}}{k_B T}$, which is indeed the same for every reaction.

We follow the slightly unusual convention of treating $A_i + A_j \rightleftharpoons A_{i+j}$ and $A_j + A_i \rightleftharpoons A_{i+j}$ as two distinct reactions when $i \neq j$. This is merely a convenience that allows us to write the equations below in a more compact form. (Without this, reactions of the form $2A_i \rightleftharpoons A_{2i}$ need to be treated as a special case, since there are two distinct bonds that can be broken to split an A_5 into an A_2 and an A_3 , but only one way to split A_4 into two A_2 dimers.)

If every reaction of the form in Equation 1 is included in the network then the results are not especially interesting. However, we can construct A-polymerisation chemistries with more complex dynamics by disallowing particular reactions. In real chemistry reactions may be disallowed (or happen at very slow rates) for a variety of reasons, such as conservation laws or steric effects. Our goal here is not to model such effects specifically, but rather to understand, in general, the effects of imposing constraints on the reaction network.

Let $r_{i,j} = 1$ if the reaction $A_i + A_j \rightleftharpoons A_{i+j}$ is included in the network, and 0 otherwise, with $r_{i,j} = r_{j,i}$. Mass action kinetics then leads to the ordinary differential equations

$$\frac{da_i}{dt} = \sum_{j=1}^{i-1} r_{i,j-i} (k_{s} a_j a_{i-j} - k_{d} a_i)
+ \sum_{j=1}^{\infty} (r_{i,j} + r_{j,i}) (k_{d} a_{i+j} - k_{s} a_i a_j) + \phi_i,$$
(2)

where a_i represents the concentration of A_i . The term $k_s a_j a_{i-j}$ represents the formation of A_i through the joining of A_j to A_{j-i} ; the $k_d a_i$ term is the loss of A_i through splitting into A_j and A_{j-i} ; the $k_d a_{i+j}$ term is the gain in A_i due to A_{i+j} splitting into A_i and A_j ; and $k_s a_i a_j$ represents the loss of A_i by joining to A_j to make A_{i+j} .

The ϕ_i terms represent the flux of each species A_i in or out of the system. These terms may be set to non-zero values in order to model an open system with external energy sources. In most of this paper these terms will be set to zero, representing a closed system whose free energy comes purely from a non-equilibrium initial state.

The linear case: self-organisation of first-order autocatalysis

In this section we present a simple non-equilibrium A-polymerisation chemistry that can be solved semi-analytically. This gives some insight into some sufficient conditions for exponential growth, and the reasons why it emerges under these conditions.

In this chemistry we set $r_{1,1}=0$ and all other $r_{i,j}=1$. That is, two monomers (A_1) are not allowed to directly join to form a dimer (A_2) and, *vice versa*, a dimer cannot directly split into monomers. All other reactions are permitted. There are multi-step reactions that can convert monomers

into dimers in this system, such as the autocatalytic cycle

$$\begin{aligned} \mathbf{A}_1 + \mathbf{A}_2 &\longrightarrow \mathbf{A}_3 \\ \mathbf{A}_1 + \mathbf{A}_3 &\longrightarrow \mathbf{A}_4 \\ \mathbf{A}_4 &\longrightarrow 2\,\mathbf{A}_2, \end{aligned} \tag{3}$$

which has the net reaction $2A_1 + A_2 \longrightarrow 2A_2$. (Here, as below, we use unidirectional arrows to represent the net direction in which these reversible reactions proceed.) This is called a first-order autocatalytic cycle, because there is only one A_2 on the left-hand side. First-order autocatalysis leads to exponential growth, as we will see below. (See King (1978, 1982) for background on the theory of autocatalytic cycles, and Andersen et al. (2012) for a formal definition.)

Let us suppose that the initial conditions contain a very high concentration of A_1 , and trace amounts of every other species. That is, a_1 is large and a_i very small for i>1. Under these assumptions, the rate of change of a_1 will be small in comparison to its value, so that we can treat it as constant. Using the approximation $a_ia_j=0$ for i,j>0, Equations 2 become

$$\frac{da_2}{dt} = 2\sum_{j=1}^{\infty} (k_d a_{2+j}) - 2k_s a_1 a_2$$

$$\frac{da_i}{dt} = 2k_s a_1 a_{i-1} - (i-1)k_d a_i$$

$$+ 2\sum_{j=1}^{\infty} (k_d a_{i+j}) - 2k_s a_1 a_i$$
(4)

for i > 2. This may be written $d\mathbf{a}/dt = R\mathbf{a}$, where \mathbf{a} is a column vector with elements a_2, a_3, \ldots , and R is the matrix

$$k_{d} \begin{pmatrix} -2K & 2 & 2 & 2 & \cdots \\ 2K & -2K - 2 & 2 & 2 & \cdots \\ 0 & 2K & -2K - 3 & 2 & \cdots \\ 0 & 0 & 2K & -2K - 4 & \cdots \\ \vdots & & & \ddots \end{pmatrix},$$
(5)

with $K = a_1 k_s / k_d$.

It should be noted that although this approximation leads to a linear system of equations, it is not a "near-equilibrium" approximation in the thermodynamic sense. The thermodynamic equilibrium of this type of system will in general contain a positive concentration of every species. We are linearising the system around the state where most of the species are at zero concentrations; this state is a dynamical fixed point, but it is very far from thermodynamic equilibrium.

Since this is a linear dynamical system, its asymptotic behaviour is determined by the eigenvectors and eigenvalues of R. In particular, since the only negative elements of R are on the diagonal, R has only one eigenvector \mathbf{v} with all positive entries (by application of the Perron-Frobenius theorem to

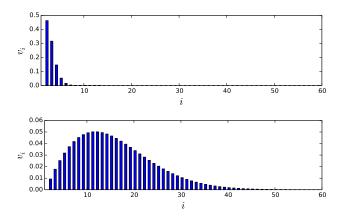


Figure 1: The elements of the leading eigenvector of the matrix R, for K=1.0 (top) and K=100.0 (bottom). The corresponding eigenvalues depend on $k_{\rm d}$, but are positive in both cases, meaning that the concentrations grow exponentially while remaining proportional to the elements of the leading eigenvector. Both represent exponential growth, but through different mechanisms.

the irreducible matrix $e^{\varepsilon R} \approx I + \varepsilon R$ for sufficiently small ε), whose corresponding eigenvalue λ is real. For sufficiently large t we have $a_i(t) \approx A v_i e^{\lambda t}$, with A a real constant that depends on the initial conditions. A=0 in the case where every $a_i=0$ (for i>1). For all other initial conditions, A is positive.

The eigenvector and corresponding eigenvalue can readily be found by power iteration of $e^{\mathbf{R}^{(N)}}$, where $\mathbf{R}^{(N)}$ is an $N \times N$ matrix whose entries are the same as those of \mathbf{R} (with the exception that $\mathbf{R}^{(N)}_{NN} = -N$, due to the absence of the reaction $\mathbf{A}_N + \mathbf{A}_1 \longrightarrow \mathbf{A}_{N+1}$. For sufficiently large N this makes very little difference).

Figure 1 shows two examples. In both, the leading eigenvalue of R is positive, meaning that the species other than A_1 will grow exponentially. In the case where K=1, this growth is mostly due to the minimal autocatalytic cycle shown in Equation 3, with a few side reactions producing longer polymers.

However the case with a larger K (meaning proportionally faster reaction rates in the synthesis direction, or just a higher concentration of A_1 initially) is dominated instead by larger molecules, and its growth is therefore due to much larger autocatalytic cycles. There is no one cycle that dominates; instead the microscopic picture is of a collection of polymers of many different lengths, which grow at their ends due to monomer addition, but which also occasionally split somewhere in the middle, giving rise to two new polymers.

It is a theorem in mass action kinetics that there is a global Lyapunov function given by the Gibbs energy, whose minimum is at the thermodynamic equilibrium point. Since we are linearising around a different fixed point, this guarantees that it will not be stable; at least one of its eigenvalues must be non-negative. Because of this, there are no parameters that have to be tuned in order to observe autocatalysis in this model. For any value of K there will always be a region around the point $a_2=a_3=\cdots=0$ in which autocatalytic growth occurs.

To derive these results we assumed that a_i was small for i > 1, but since the concentrations of these are increasing exponentially, this assumption will be violated eventually. This means that either a_1 will start to decrease at a substantial rate, changing the value of K, or terms of the form $a_i a_j$ will start to be important, changing the dynamics. In a real system, this may or may not happen before the exponential growth becomes manifest enough to be clearly observable.

It is worth examining the assumptions we used to obtain this result. One important part of the model is that the reaction $2A_1 \rightleftharpoons A_2$ is removed from the reaction network. The reason for this is that this reaction would rapidly produce enough A_2 that the assumption of small a_2 would be violated. We propose that in general the absence of such a "direct route" to product formation will be necessary in order for autocatalysis to be observed.

The other important assumption was that the polymer species have (initially) very low concentrations, so that the a_ia_j terms vanish. Physically, this corresponds to a solution so dilute that two polymer molecules are vanishingly unlikely to meet and react. (A₁ monomers, however, are highly concentrated, so the probability of reacting with a monomer is high.) Under these conditions, exponential growth can occur through the molecules (a) "growing" into larger molecules by reacting with monomers, and (b) spontaneously splitting into two smaller molecules.

Such dynamics can be expected in much more complex thermodynamically reasonable chemistries, as long as these conditions are obeyed; our reasoning is applicable to a much broader class of models than the one presented here. The chemistry need not take the form of A-polymerisation; this reasoning will hold as long as there is some set of "food" species that cannot directly react to form products, and so long as there are no energetically irreversible "sinks" in the network. No autocatalytic cycles can arise unless they already exist in the network of potential reactions, but our example shows that for even such a simple chemistry as A-polymerisation there are many such potential loops. We suspect these conditions will be satisfied by many chemistries, as long as they are sufficiently densely connected and all the reactions have some degree of reversibility.

The mechanism as described so far results in *exponential* growth. Equations 4 are a linear dynamical system, and therefore the only possibilities are exponential growth, exponential decay or neutral stability. (Oscillation around the equilibrium point is impossible, because it would require concentrations to become negative.) The more general assumptions outlined in the previous paragraph will also lead

to linear dynamical systems, resulting either in exponential growth or decay. In the next section we explore the importance of non-linearity in chemical networks, and we demonstrate that with some changes to the reaction network, self-organisation can produce super-exponential (or "hyperbolic") growth in a simple A-polymerisation chemistry.

Non-linearity and super-exponential growth

The constituents of a living cell are not capable of exponential growth at low concentrations; one of the many functions of the cell membrane is to keep the concentrations high enough for growth to occur. If the components of a living cell were to be placed in a dilute solution, it would be unlikely for an enzyme to collide with its substrate in order to bind to it, and therefore enzyme-catalysed reactions would occur at vanishingly low rates.

Exponential autocatalysis is interesting from an origins of life point of view (Virgo and Ikegami, 2013), precisely because it doesn't require high concentrations, and therefore could exist before the evolution of the cell membrane or other forms of compartmentalisation. However, the linear analysis in the previous section makes it clear that the concentrations will always converge towards the *same* profile, regardless of the initial conditions (as long as they obey the low concentration assumption). This seems to leave no room for evolution: there is no way one exponential replicator can out-compete another.

Many models of prebiotic chemistry are nonlinear, because they involve reactions where two non-food molecules must collide with each other. These include all models based on explicit catalysis, such as the work of Kauffman (1986) or the hypercycle model of Eigen and Schuster (1979).

For these reasons it is worth exploring under which circumstances non-linear, super-exponential growth can emerge spontaneously. In this section we demonstrate circumstances in which this can happen in A-polymerisation chemistries. The key is to constrain the reaction network such that first-order, exponential autocatalytic cycles cannot occur. Under these circumstances, growth occurs if we assume reactions of the form a_ia_j (for i,j>0) cannot occur. However, when integrating the model numerically, we find autocatalytic cycles that rely on these reactions. Their kinetics are non-linear (in fact super-exponential), because they involve these quadratic terms. We show that this non-linearity can give rise to behaviours such as bistability and oscillations.

The exponential growth in the previous section occurs because a molecules can split into two shorter molecules, each of which can then grow through monomer addition until it reaches the length of the original molecule, giving a net reaction $A_n + nA_1 \longrightarrow 2 A_n$. Now we ask what happens if we prevent this simple first-order autocatalysis mechanism while retaining many of the possible reactions from the original network. We show that autocatalysis still self-organises,

but it does so through a more complex, second-order mechanism that results in super-exponential growth.

In order to prevent this, we constrain the reaction network in the following way: we define a set of "banned species" B. A reaction $A_i + A_j \Longrightarrow A_{i+j}$ is included in the reaction network *unless* either $A_i \in B$, $A_j \in B$, or $A_{i+j} \in B$.

This technique of "banning" species is merely a convenient way to prevent the formation of first-order autocatalytic cycles while retaining much of the original network; it is not supposed to directly represent something one would expect to happen in polymerisation chemistry. However, in more complex chemistries a species may be stiochiometrically possible but have such a high free energy that it is never formed, and so networks with broadly this type of structure might not be entirely absent from real chemistry. We suspect that there are also many other reasons why a complex network would fail to contain first-order loops.

To see how banning species can prevent first-order autocatalysis, let us imagine that $B=\{A_{2^{i+1}}:i\in\mathbb{Z}\}$. That is, molecules whose lengths are powers of two are banned, apart from A_1 . A molecule of A_n (with n not a power of two) may split into two smaller molecules, A_m and A_{n-m} , as long as neither m nor n-m is a power of two. However, at least one of these molecules is no longer able to grow by monomer addition until it reaches length n. For let $r=2^{\lfloor \log_2 n \rfloor}$ be the largest power of two less than n. Then either m < r or n-m < r, since $r \ge n/2$. A molecule of size less than r cannot grow to size n by monomer addition alone due to the lack of the reaction $A_{r-1}+A_1 \longrightarrow A_r$. Therefore net reactions of the form $A_n+nA_1 \longrightarrow 2A_n$ are impossible in this chemistry.

By excluding even more species we can obtain quite complex behaviour. Here we present results from a system that excludes the species $B = \{A_{3i} : i \in \mathbb{Z}\}$, and also excludes the reaction $2A_1 \rightleftharpoons A_2$. For computational reasons we also exclude species longer than 60 monomer units.

If, as in the previous section, we linearise this system around the point where all non-food species have zero concentration, we find that the leading eigenvalue is zero. In such cases the stability of the fixed point is determined by the nonlinear terms, according to neutral manifold theory. Since we know there is a global Lyapunov function (the Gibbs energy) with a minimum at a different point, we should expect the nonlinear terms to result in a second-order instability around this fixed point. Our numerical results in this section show that this is indeed the case.

Figure 2 shows the results of numerically integrating this system, using Equations 2 with no approximations, from an initial condition consisting of a high concentration of A_1 and small amounts of everything else. The result is that at around t=450 there is a sudden transition to the equilibrium state, in which all species are present, exponentially distributed according to their lengths. These dynamics indicate that the species other than A_1 grow (super-)exponentially at low

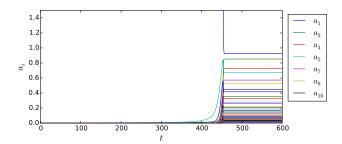


Figure 2: The results of numerically integrating an A-polymerisation chemistry over time. This chemistry excludes the species $B = \{A_{3i} : i \in \mathbb{Z}\} \cup \{A_i : i > 60\}$, and in addition excludes the reaction $2A_1 \Longrightarrow A_2$. The parameters used are $k_s = k_d = 1$. The initial concentration of A_1 is 100, and the initial concentrations of all other species are 10^{-5} . Prior to the sudden transition at around t = 450, the species' concentrations grow super-exponentially through second and third-order autocatalysis.

Reaction	Rate
$A_4 + A_1 \longrightarrow A_5$	3.194×10^{-5}
$A_7 + A_1 \longrightarrow A_8$	1.888×10^{-5}
$A_8 \longrightarrow 2 A_4$	1.882×10^{-5}
$A_4 \longrightarrow 2 A_2$	1.366×10^{-5}
$A_5 + A_2 \longrightarrow A_7$	1.084×10^{-5}
$2 A_5 \longrightarrow A_{10}$	8.050×10^{-6}
$A_{10} + A_1 \longrightarrow A_{11}$	8.021×10^{-6}
$A_{11} \longrightarrow A_7 + A_4$	7.942×10^{-6}
$A_{13} + A_1 \longrightarrow A_{14}$	9.351×10^{-8}
$A_{10} \longrightarrow A_8 + A_2$	8.598×10^{-8}

Table 1: The top ten reactions in the system shown in Figure 2, at time 100. The rates shown are absolute net rates, in moles per unit time.

rates until their concentrations become high enough for the growth's effects to be felt. This is similar to the results from first-order autocatalytic systems presented in (Virgo and Ikegami, 2013). However, the mechanism is rather more complicated.

At time t=100, the growth in the system is largely due to the following set of reactions:

$$\begin{array}{c} A_{8} \longrightarrow 2\,A_{4} \\ A_{4} + A_{1} \longrightarrow A_{5} \\ A_{4} \longrightarrow 2\,A_{2} \\ A_{5} + A_{2} \longrightarrow A_{7} \\ A_{7} + A_{1} \longrightarrow A_{8}. \end{array} \tag{6}$$

These reactions may be stiochiometrically balanced to form the net reaction $3 A_8 + 8 A_1 \longrightarrow 4 A_8$. It can therefore be classified as third-order autocatalysis. (Though these are not the only reactions in the system, and they are not necessarily happening at stoichiometrically balanced rates, because the

Reaction	Rate
$A_4 + A_1 \longrightarrow A_5$	3.745×10^{-3}
$A_7 + A_1 \longrightarrow A_8$	1.549×10^{-3}
$A_8 \longrightarrow 2 A_4$	1.429×10^{-3}
$A_{10} + A_1 \longrightarrow A_{11}$	1.202×10^{-3}
$2 A_5 \longrightarrow A_{10}$	1.150×10^{-3}
$A_{11} \longrightarrow A_7 + A_4$	1.088×10^{-3}
$A_5 + A_2 \longrightarrow A_7$	3.669×10^{-4}
$A_4 \longrightarrow 2 A_2$	2.466×10^{-4}
$A_{13} + A_1 \longrightarrow A_{14}$	7.950×10^{-5}
$A_8 + A_5 \longrightarrow A_{13}$	3.669×10^{-5}

Table 2: The top ten reactions in the system shown in Figure 2, at time 400. The overall difference in magnitudes compared to Table 1 is due to increases in concentrations over time.

concentrations are changing over time. See Table 1 for the true reaction rates.) The nonlinearity comes from the reaction $A_5 + A_2 \longrightarrow A_7$, which requires an A_5 to collide with an A_2 dimer in order to proceed; it will therefore increase in rate as the concentrations of these species increase.

By t=400 the system is dominated by the following overlapping but different set of autocatalytic reactions, as can be seen in Table 2:

$$A_{11} \longrightarrow A_7 + A_4$$

$$A_7 + A_1 \longrightarrow A_8$$

$$A_8 \longrightarrow 2 A_4$$

$$A_4 + A_1 \longrightarrow A_5$$

$$2 A_5 \longrightarrow A_{10}$$

$$A_{10} + A_1 \longrightarrow A_{11}.$$

$$(7)$$

This is a second-order autocatalytic cycle with the net reaction $2\,A_{11}+11\,A_1\longrightarrow 3\,A_{11}$. Where the cycle in Equations 6 produces A_7 by splitting A_4 into A_2 and then joining it with A_5 , this network instead produces A_7 by forming and then splitting A_{11} . The transition from Equations 6 to 7 is probably the result of the general increase in concentrations, making the reaction $2\,A_5\longrightarrow A_{10}$ more likely to proceed. In this network, $2\,A_5\longrightarrow A_{10}$ is the nonlinear step.

We reiterate that this nonlinear autocatalytic increase in concentrations is a logical consequence of the reaction network's structure. The values of the numerical parameters are more or less arbitrary; the only tuning required is to set the initial concentrations small enough for the effect to be observed.

This nonlinearity can lead to bistability. Suppose that additional reactions are added such that the polymer species decay exponentially into inert products. Then at low concentrations the decay reactions will be faster than the autocatalytic cycles and the polymer species will not be able to persist in the system. However, at higher concentrations the autocatalytic species will grow at a faster rate than their decay. Thus the polymer species can persist in the system

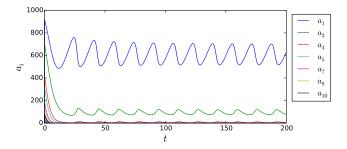


Figure 3: Convergence to a limit cycle due to a higher-order autocatalytic cycle in a flow reactor. The reaction network is the same as for Figure 2. The flow reactor is modelled with the flux terms $\phi_1=0.4~(10000-a_1)$ and $\phi_i=-0.01a_i$ for i>1. The parameters used are $k_s=1.0$ and $k_d=1200$. The autocatalysis is primarily due to the reactions shown in Equation 6.

only if they are initially present in concentrations above a certain threshold. (The technical details of this result are not especially enlightening, so we omit them here.) This is in contrast to the case of exponential growth, where the zero-concentrations state is an unstable fixed point, meaning that even a single molecule can eventually grow to dominate the system.

The existence of a concentration threshold leads to a kind of "precariousness" that is an important property of life-like systems, since it is analogous to biological death (Virgo, 2011; Barandiaran and Egbert, 2013). Precariousness means there are some perturbations from which a system cannot recover; in this case some perturbations push the concentrations below the threshold, and the concentrations must then eventually decay to nothing.

This same kind of nonlinearity can also lead to a more easily visualised phenomenon, namely temporal oscillations, as shown in Figure 3. To obtain this result, the reaction is modelled as if in a flow reactor. There is a constant input of A_1 and exponential decay of every other species, which can be thought of as the species leaving the system in the reactor's outflow. The oscillations occur through a similar mechanism to the famous lynx-hare cycle in ecology: the autocatalytic cycle consumes too much of its food and as a result its constituent species decay somewhat. This gives the food concentration a chance to recover (since it is being continually fed in to the system), and the autocatalytic species once again increase in concentration, completing the cycle.

This oscillatory behaviour does not occur for every choice of parameter values. However, it demonstrates nicely that the nonlinearity caused by higher-order autocatalysis can lead to phenomena that are not possible in the linear, first-order case. We reiterate that all of the other results in this paper are very robust and require no parameter tuning at all.

Analysis and Conclusions

The second law of thermodynamics implies that physical systems must always "flow downhill", where "downhill" means in the direction of increasing entropy, or decreasing free energy. We have implemented this feature in a very simple class of chemical reaction networks and found that it can lead to self-organisation of complex autocatalytic cycles.

This is in some way analogous to the self-organisation of dissipative structures in physical systems, such as convection cells. These also involve the formation of cycles, and also increase the rate at which energy is dissipated.

Our informal, intuitive understanding of these results is as follows: under some (but by no means all) circumstances, physical systems far from equilibrium not only "flow downhill", but also tend to seek out faster and faster ways to do this. The formation of convection cells is again an example of this; the breaking of a dam is perhaps a more vivid one. The "full" A-polymerisation reaction network (with all reactions included) will flow to its equilibrium rather rapidly. However, if we block the most direct route (the $2A_1 \rightleftharpoons A_2$ reaction) then an indirect route must be found, by taking advantage of catalytic cycles. Blocking the zeroth-order route leads to the self-organisation of first-order autocatalytic cycles, because (in a very informal sense) these lead to faster rates of dissipation.

Blocking these first-order solutions then leads to higherorder solutions. It is as if the system tries to find the easiest route towards its equilibrium, and only once the simplest paths are blocked does it find the more complex ones. Having tried to find autocatalytic networks in these chemistries using pen and paper, the solutions found by the simulation seem rather elegant and novel.

From a more formal point of view, these results occur because the initial conditions are near a fixed point that is not the thermodynamic equilibrium point, and therefore cannot be stable. By preventing first-order instability, we force this fixed point to exhibit a second-order instability.

The nonlinearity of the resulting systems leads to phenomena such as precariousness and oscillations. A nonlinear system with many equilibria is more interesting from an origins of life point of view, because it is closer to a system in which multiple different autocatalytic systems can compete with one another, leading to evolution.

It is interesting to note that we found complex behaviour in this system by removing reactions rather than by adding them. If we consider the full A-polymerisation chemistry, including both forward and reverse reactions, all of these autocatalytic cycles — both first and higher-order — are already present as its subnetworks. Whether the more complex networks become manifested or not is a matter of whether the right constraints exist to prevent simpler networks from being manifested instead.

In this respect we are reminded of Terrence Deacon's (2011) comments about constraints in relation to the origins

of life. Our systems do not yet go as far as Deacon's ideas, since the constraints are imposed from outside rather than being generated by the system itself, but nevertheless it is an interesting demonstration of the idea that constraints can lead to complexity.

Many authors in the origins of life have worried about the problem of "side-reactions" that can prevent autocatalysis from occurring (e.g. Szathmáry, 2000; King, 1982; Virgo et al., 2012). Our results suggest that under the right energetic conditions, such issues might not arise. Our reaction networks contain many possible reactions besides the ones that eventually form the autocatalytic cycles, yet these do not seem to disrupt the autocatalysis to any meaningful extent. The reason is that any molecules produced by side reactions eventually end up participating in autocatalytic cycles of their own.

A closely related issue in the origins of life is the combinatorial explosion that can arise from several small molecular components combining in different ways. Our model is incomplete in regards to this issue — we have only one type of monomer, and they combine only linearly — but nevertheless our results suggest that the combinatorial explosion might not be as serious an issue as it might at first seem. One can imagine a class of molecules that can grow in multiple ways, leading to a combinatorial explosion, but which can also undergo reactions that decompose them into smaller molecules. With such a chemistry, at low concentrations one should still expect exponential growth through the mechanism that arises in the first of our models. We therefore suggest that the future of origins of life models might be not in "taming the combinatorial explosion" (Schuster, 2000) but in embracing it. If every molecule forms part of an autocatalytic cycle then the combinatorial explosion represents an efficient way to explore the space of possible replicators, and it may be that it was through such a process that the first high-fidelity replicators arose.

This research represents a step towards understanding what properties a chemical reaction network must have in order for complex, life-like dynamics to emerge. There is still a lot of work to do to understand the effects of network topology and energetics on the emergence of complex dynamics, but the results in this paper represent a substantial step in the right direction.

References

- Andersen, J. L., Andersen, T., Flamm, C., Hanczyc, M., Merkle, D., and Stadler, P. F. (2013). Navigating the chemical space of hcn polymerization and hydrolysis: Guiding graph grammars by mass spectrometry data. *Entropy*, 15:4066–4083.
- Andersen, J. L., Flamm, C., Merkle, D., and Stadler, P. (2012). Maximizing output and recognizing autocatalysis in chemical reaction networks is np-complete. *Journal of Systems Chemistry*, 3(1):1–9.
- Barandiaran, X. and Egbert, M. (2013). Norm-establishing

- and norm-following in autonomous agency. *Artificial Life*, 20(1):5–28.
- Deacon, T. (2011). *Incomplete Nature: How Mind Emerged from Matter*. W. W. Norton & Company.
- Eigen, M. and Schuster, P. (1979). *The Hypercycle: A principle of natural self-organisation*. Springer.
- Fontana, W. and Buss, L. W. (1994). What would be conserved if "the tape were played twice"? *Proceedings of the National Academy of Sciences*, 91:757–761.
- Kauffman, S. (1986). Autocatalytic sets of proteins. *Journal of Theoretical Biology*, 119:1–24.
- King, G. A. M. (1978). Autocatalysis. *Chemical Society Reviews*, 7(2):297–316.
- King, G. A. M. (1982). Recycling, reproduction and life's origins. *BioSystems*, 15:89–97.
- Kondepudi, D. and Prigogine, I. (1998). Modern Thermodynamics: from heat engines to dissipative structures. Wiley.
- Schuster, P. (2000). Taming combinartorial explosion. *PNAS*, 97(14):7678–7680.
- Szathmáry, E. (2000). The evolution of replicators. *Philosophical Transactions of the Royal Society, Series B.*, 355:1669–1676.
- Vasas, V., Fernando, C., Santos, M., Kauffman, S., and Szathmáry, E. (2012). Evolution before genes. *Biology Direct*, 7(1).
- Virgo, N. (2011). Thermodynamics and the Structure of Living Systems. PhD thesis, University of Sussex, UK.
- Virgo, N., Fernando, C., Bigge, B., and Husbands, P. (2012). Evolvable physical self-replicators. Artificial Life, 18:129–142.
- Virgo, N. and Ikegami, T. (2013). Autocatalysis before enzymes: The emergence of prebiotic chain reactions. In Advances in Artificial Life, ECAL. MIT Press.

Computational novelty: Phenomena, mechanisms, worlds

Adam Nellis Susan Stepney

York Centre for Complex Systems Analysis, University of York, UK, Y010 5DD adam.nellis@york.ac.uk susan.stepney@york.ac.uk www.yccsa.org

Abstract

We want to implement a variety of computer programs capable of generating unbounded novelty via emergent evolution. We also want a principled, structured way of analysing our novelty-generation programs and improving them over time.

We have developed a definition of *embodiment* in terms of a conceptual framework of hierarchical *phenomena*, *mechanisms*, *and worlds*, for use in analysing and building novelty-generation programs. We present these concepts and demonstrate them on two existing artificial chemistry systems: Stringmol and GraphMol. Our two demonstration systems behave in very different ways. We describe these differences, explaining why they arise in terms of our framework. The systems better able to generate novelty have more embodied implementations.

Keywords: autopoiesis, emergence, embodiment, evolutionary algorithms, bio-inspired algorithms, artificial chemistry

Introduction

One of the aims of Artificial Life is the construction of computational artefacts that behave like the living organisms we observe in nature. Rather than creating computational versions of *specific* living organisms, we aim to capture the *general properties* displayed by living organisms (and not displayed by current computers), and replicate these within computational substrates. The most striking difference between the living world and the computational world is the creative process of biological evolution. The living world creates novel organisms that solve life's problems in new and interesting ways. Replicating this novelty-generation process within a computer program is one of the ultimate goals of Artificial Life.

Here we talk about computer programs generating novelty through emergent evolution. We use the concepts of embodiment within virtual worlds and emergent properties of computer programs. We present a new definition of virtual *embodiment*, and a language to talk about different novelty generation systems, in terms of emergent *phenomena*, their underlying *mechanisms*, and the *world* within which they are embodied. We end by applying these to two existing

example novelty-generation systems, Stringmol and Graph-Mol, to demonstrate the different emergent consequences of alternative embodiments of the same mechanisms.

Computational novelty generation

Can computers perform tasks that have not been programmed into them? Can they invent their own, novel, behaviours, overriding their original programming and performing tasks of their own devising? A famous quote by Ada Lovelace cautions: "The Analytical Engine has no pretensions whatever to *originate* any thing. It can do whatever we *know how to order it* to perform." (Lovelace, 1843) [italics in original].

Evolutionary computation is attempting to do what Lovelace stated as impossible. It explicitly tries to develop computer programs that can originate their own solutions, and use them to solve problems that we *do not know how to solve*. Why would we imagine it could be possible to develop such programs? Because such a process already exists: *biological evolution*.

The evolution of life has produced a staggering variety of forms. Darwin encapsulated the idea of biological evolution producing novel things that are different from those it started with: "...from so simple a beginning endless forms most beautiful and most wonderful have been, and are being, evolved." (Darwin, 1859, p.429).

We can construct a computer simulation of biological evolution, by programming our computer to perform the rules laid out in Darwin's seminal work. We might then expect this computer program to display *endless forms most beautiful and most wonderful*, as does biological evolution. It is possible to write such a computer program *in principle*, but the *practice* of actually implementing it is very difficult.

We have Lovelace's dictum that computer programs can do only what we know how to order them to do. Then how might a computer program produce novel things that we did not explicitly program into it? The answer is *complex systems*.

Complex systems have simple rules but complicated behaviours that cannot be predicted without observing the system. There are some computer programs that are very easy to code, but that have behaviour impossible to predict without running the program. We know *how to do* the computation, but we cannot know the *result* without running the computation explicitly. Thus we can order our computer programs to perform tasks that we do not know (and *cannot know*) the outcomes of beforehand.

Biological evolution is a complex system. We can know the simple rules by which biological evolution operates on a day-to-day basis, but this knowledge does not allow us to predict the *endless forms* that will be produced far into the future. The challenge for evolutionary computation is working out which simple rules to order our computer programs to perform, so that we can observe them generating endless *virtual* forms. Mere enumeration of such programs is clearly insufficient: it would result in a useless Library of (Evolutionary) Babel.

Emergent evolution

Traditional evolutionary algorithms such as Genetic Algorithms (Holland, 1992; Goldberg, 1989) and Genetic Programming (Cramer, 1985; Koza, 1992) involve representing an engineering problem as a point in a pre-defined multidimensional space, and then searching this space using analogues of biological mutation, crossover and selection. Each candidate solution to the problem (organism) is assessed for its *fitness*, by testing how well it solves the engineering problem. The fittest organisms survive to reproduce with variation, producing the next generation of organisms.

These algorithms are computational models of biological evolution, and in some cases they have produced interesting solutions to engineering problems that humans would not have designed. But they have a problem: they do not produce Dawins's *endless forms* in their search through the static pre-defined space. In current evolutionary algorithms, there is a clear distinction between *the organisms being evolved* and *the algorithm evolving them*. The algorithm is a fixed computer program, and the organisms are the changing solutions to an engineering problem.

A more biological perspective sees the 'algorithm' of biological evolution intertwined with the organisms it is evolving. Algorithm and organisms co-evolve and co-create each other (Kauffman, 2000). As well as the organisms changing (evolution), the 'evolutionary algorithm' changes (metaevolution), changing the very way in which the organisms evolve. The evolution of new organisms leads to the evolution of new ways of evolving new organisms. These are the kinds of dynamics we seek for novelty-generation.

Biological domain

There is no biological analogue of computer programs that directly 'implement evolution'. Biological evolution is an *emergent property* of a system of interacting parts (organisms in an environment). When evolution changes individ-

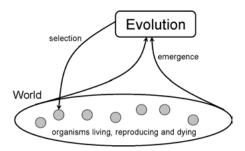


Figure 1: Organisms create evolution: evolution is an emergent property of a system of organisms, so evolution acting on those organisms can itself be changed.

ual parts of the system, it changes the system as a whole. This in turn changes the emergent properties of the system, one of which is biological evolution (figure 1). When viewed as an emergent property, it is difficult to see how biological evolution can change the organisms of a system and *not* change its own 'algorithm'.

The different parts comprising biology (organisms, reproduction, death, etc.) are not separate, isolated units, akin to functions in a computer program. They are all 'implemented in' the same language: that of *chemistry*. When evolution selects biological organisms, it is selecting collections of chemicals. The whole of biology emerges from these chemicals, so selection has the power to change what types of biology are seen (for example, creating new niches for new organisms, or sending species extinct). Evolution emerges from biology, so selection also has the power to change what types of evolution are seen. It is this closure of the loop that gives biological evolution the power of meta-evolution: implementing the processes that lead to evolution in the same language as the organisms that are being evolved.

Computational platform

The idea of *evolving an evolutionary algorithm* can encompass many different computational models. But, to date, noone has produced a landmark example of a successful model of computational *meta*-evolution.

Our biological domain model above specifies a sub-space of possible computational models. Rather than thinking about the general task of evolving an evolutionary algorithm, we focus on the more specific task of making evolution an emergent property of virtual organisms interacting in a virtual world. We require a virtual world in which we can:

- 1. Implement virtual organisms capable of solving an application problem (which might be just survival).
- 2. Implement the components of evolution: reproduction of the organisms, competition for resources, and death.
- 3. Encode the above features within the genomes of the virtual organisms themselves, so that these too can be evolved.

 Do all this in an evolvable way, so that evolution will be able to move through the space of organisms and evolutionary algorithms.

Classic ALife systems tend to implement features 1 and 2 (evolution), but in a non-evolvable manner, so precluding 3 and 4 (meta-evolution). Here we describe some concepts useful for designing virtual worlds that support meta-evolution, and then present two different but related algorithms that start to explore the space of these virtual worlds.

The idea underlying emergent evolution is: By implementing evolution as a process in terms of a suitable (evolvable) lower-level world, we gain a benefit: meta-evolution. We build on the ALife view of autopoiesis, and the computer science view of embodiment.

Autopoiesis

Autopoiesis is Maturana and Varela's theory of lifeforms (either physical or virtual) being embodied within worlds (Maturana and Varela, 1980; Varela, 1991). Concepts from autopoiesis help with the problem of how to design virtual worlds.

In our virtual worlds we do not have the vast resources of the natural world, so we ask: what are the features of the natural world that make it amenable to emergent evolution? And: how can we replicate analogous features within computer programs? Autopoiesis goes a way towards answering the first of these questions, and provides a framework for exploring the second. Varela says that:

"Autopoiesis is [an interplay] between the local component levels and the global whole, linked together in reciprocal relation through the requirement of constitution of an entity that self-separates from its background." (Varela, 1991, p.84)

So our virtual organisms must be an emergent property: not implemented as basic units in the world, but composed of the world's basic units interacting to define an organism as a self-sustaining property of the world.

Varela talks about the low-level components of the world, and the "surplus of signification" these components take on in the context of an organism:

"There is no food significance in sucrose except when a bacterium swims upgradient and its metabolism uses the molecule in a way that allows its identity to continue. This surplus is obviously not indifferent to the regularities and texture (i.e., the "laws") that operate in the environment, that sucrose can create a gradient and traverse a cell membrane and so on. On the contrary, the system's world is built *on* these regularities, which is what assures it can maintain its coupling at all times." (Varela, 1991, p.86)

So our virtual world must contain sufficient structure for the organisms to use and exploit, and endow with a *surplus of*

signification. We call this *richness*: a property of our virtual world is *rich* if it can vary in different ways, and organisms can control this variation, making use of it in different ways to sustain themselves.

If a world is rich enough, then organisms can choose how to construct a living out of it. There is no "one right way" of making a living in a rich world: there are multiple alternatives that organisms construct. For different types of organism to coexist, not only there is enough resource to share between them, also they are able to use the same resources in different ways and for different purposes. A rich world is one that allows these different purposes and different organisms to co-construct each other.

Embodiment

Quick was the first to apply the ideas of embodiment to worlds other than the physical world (Quick et al., 2000; Stepney, 2007). We are talking about creating a virtual world within a computer program, and embodying virtual organisms within this virtual world. The benefits gained by these organisms from their embodiment are with respect to the virtual world, not the physical world. Quick defines embodiment without reference to the physical world, by expressing the process of embodiment in terms of two dynamical systems mutually perturbing each other. This gives us a set of requirements for systems and worlds where embodiment can happen without recourse to the physical world. This allows us to define virtual worlds in which we can implement a system and environment, and perturbations between the two, to obtain virtual embodiment. However, it describes a system being *modified* by its environment, rather than a system being constructed from its environment.

Varela's take on of embodiment focuses explicitly on (autopoietic) systems constructing themselves from their environment:

"...the living system must distinguish itself from its environment, while at the same time maintaining its coupling; this linkage cannot be detached since it is against this very environment from which the organism arises, comes forth." (Varela, 1991, p.85)

This provides a useful conceptual framework, however, Varela focuses specifically on *one organism maintaining itself but not evolving*.

We use the ideas of Maturana & Varela and of Quick, but cast them in a form useful for describing and designing embodied systems, systems made from the same kind of stuff as their environment, that can consequently *evolve*.

Phenomena, mechanisms, worlds

We present our own definition of embodiment, that we find a useful tool for (1) analysing novelty-generation programs and (2) suggesting ways of developing new novelty-generation programs.

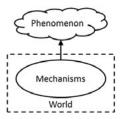


Figure 2: Our definition of embodiment: An abstract *phenomenon* (cloud) is a consequence of *mechanisms* (oval) existing in a *world* (dashed box). Mechanisms are subsets of the world, and the phenomenon is an abstract property that we can observe of the system.

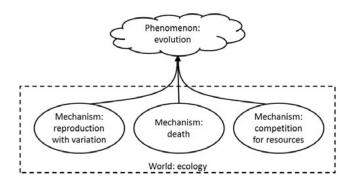


Figure 3: Emergent properties, such as evolution, can be viewed as phenomena embodied in a world by a collection of mechanisms.

We split embodiment into a *phenomenon*, some *mechanisms* and a *world*, as illustrated in figure 2. The phenomenon is an abstract concept, such as an individual reproducing. A phenomenon is embodied within a world if there exist some mechanisms within the world that enable the phenomenon to emerge.

Our definition sees the system implemented as part of the environment. There is a single world in which we identify sub-system (*mechanisms*) that we find interesting because their behaviour gives rise to an abstract phenomenon of the system. We say that the abstract phenomenon is embodied within the world, via the mechanisms. If the world contains some rich dynamics that are not part of the mechanisms, then they can influence the mechanisms.

By world, we mean a description of a system, at a particular level of abstraction. For example, the world of ecology contains entities such as animals, plants and bacteria, and mechanisms such as reproduction, growth and death; the world of string theory contains entities such as strings and forces, and mechanisms such as splitting and joining of strings.

With our definition we can talk about the same phenomenon being embodied within the same world, but via different mechanisms, or about the same phenomenon be-

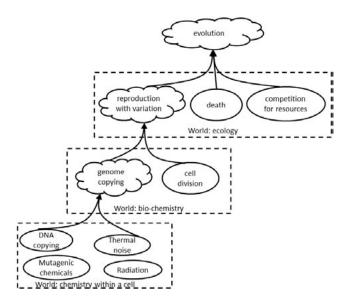


Figure 4: Embodiment is a hierarchical process, with the mechanisms on one level being the phenomena on a lower level. Identification as phenomenon or mechanism is a matter of viewpoint.

ing embodied within a different world. Defining embodiment in terms of phenomena, mechanisms and worlds allows us to consider different types of phenomenon within the same framework. For example, figure 3 shows how we can consider the emergent property of evolution to be embodied within the world of ecology via the mechanisms of reproduction with variation, death, and competition for resources. (For a more involved discussion of this definition, with examples from biology and artificial life, see (Nellis, 2012, ch.3).)

Hierarchical embodiment

When considering the embodiment of high level concepts such as evolution, we can view embodiment as a hierarchical process, figure 4. This emphasises that worlds are descriptions of a system from a particular viewpoint. We can think of evolution as the phenomenon of reproduction with variation as one of the mechanisms that embodies evolution within the biological world. But then we can also consider reproduction with variation to be an abstract phenomenon at a different level, with genome-copying and cell division as some mechanisms by which it is embodied within the biochemical world.

The mechanisms of one level are the (emergent) phenomena of the level below. Each time we move down a level of embodiment, the world becomes more detailed. For any given phenomenon, there is no one world in which it is embodied: there are multiple worlds, at different levels of detail. In terms of the physical world, going down through different levels of embodiment gives more information about

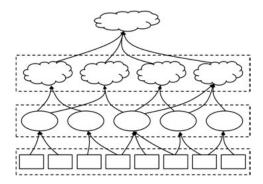


Figure 5: An ideal embodied algorithm has only one implemented world (rectangles), from which an interesting collection of mechanisms (ovals) and phenomena (clouds) emerge

how biological/chemical/physical processes happen.

Design vs. observation

When *observing* an embodied system, this hierarchical view of embodiment shows different levels at which knowledge can be constructed about a system. Different observers, focussing on different levels, will pull out different phenomena, mechanisms and worlds from the same system.

Conversely, when *designing* embodied systems as computer programs, we build these hierarchical worlds ourselves. In the biology, all the different biological mechanisms are embodied within the same world: that of chemistry and physics. This allows different biological mechanisms to interact with each other, creating a rich environment in which evolution happens. But with computer programs, this is often not the case: when designing computer algorithms, it is considered good practice to split the problem up into separate worlds that do not interact except via tightly-controlled channels. This is appropriate for traditional computer algorithms, where the sub-programs must be predictable, modular and composable. But for our requirements of novelty, connectedness and embodiment, a different approach is preferable.

For embodied computer programs, we say that two mechanisms exist in the same world if they can directly interact with each other and affect each other's behaviour. For example, in traditional genetic algorithms, the fitness function and mutation function have no direct effect on each other's behaviour, so they operate in two different worlds. But the fitness function can affect the selection function (to some extent), so they operate in the same world.

An ideal embodied algorithm would have only one world at each level, within which all the mechanisms are able to interact with each other, producing the phenomena of the level above (figure 5). But no system is ideal, and actual implementations of embodied algorithms have multiple worlds embodying different phenomena at different levels. We can build up a diagram of any embodied algorithm, showing how **Algorithm 1** Decomposing the string copy operation, as a prerequisite to embodying it

```
result string_A := string_B

i := start(string_B)
while i not at-end(string_B) do
    string_A(i) := char-copy(string_B(i))
    i := next(i)
```

many worlds its mechanisms are embodied in, and how they connect. (Two examples of these diagrams are shown later, figures 6 and 7.)

From a design perspective, the important point is to embody the particular phenomena under investigation within a richly-interacting set of mechanisms. The richer the interactions between the mechanisms, the more the mechanisms are able to change, and the more novelty-generation can be observed in the phenomenon.

Example: String copying

Species rely for their survival on individual organisms reproducing. This relies on organisms being able to copy a *template*: a string of symbols encoding the genome of the organism. As a starting point for investigating embodied evolution, we focus on *template replication* as one of the prerequisites of reproduction. All known life-forms reproduce by template replication, and no-one has yet come up with a feasible alternative for reproducing complex, evolving organisms.

A template is a string (or graph) of symbols from a small alphabet. For example, DNA sequences can be thought of as strings over the alphabet $\{A,C,G,T\}$. Computers are good at copying strings, and it is trivial to write a computer program to do so. But we do not want a *crisp* copying program that works perfectly every time and cannot change its behaviour. We want to *embody* the copying program, implementing it as a phenomenon resulting from the interaction of multiple mechanisms in a world. This will allow the copied strings to vary and evolve, and will also allow the copying *process* to vary and evolve.

Embodying string copying

We wish to embody the phenomenon of string copying in a set of mechanisms, such that their implementation makes the phenomenon emerge.

To design this as an embodied copying program, we think about the *process* of copying a string, rather than the *result* of the copy. Algorithm 1 is one way to decompose this process into parts, each involving a particular function: start, atend, char-copy, and next. Each of these four functions can be either crisp or embodied. If all four are crisp, then the overall copying process is crisp, and exact copies are always produced.

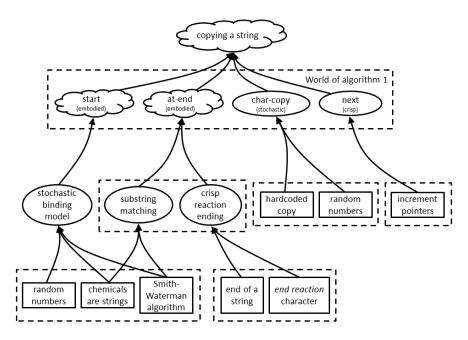


Figure 6: The mechanisms and worlds used by Stringmol to embody the phenomenon of copying a string.

We can embody the copying process in different ways, and to different degrees, by implementing virtual worlds where different combinations of these functions are embodied in different ways. We do not need to embody all four of the functions within the same world: we can implement some as crisp functions, we can embody some in simple ways, and we can embody some in complicated ways. Each of the combinations leads to different systems with different degrees of embodiment and novelty generation.

GraphMol and Stringmol are two different implementations of this embodied computational model of novelty generation. Stringmol is introduced in (Hickinbotham et al., 2010a, 2011), and described in detail in (Hickinbotham et al., 2010b). GraphMol is introduced in (Nellis and Stepney, 2011), and described in detail in (Nellis, 2012). Stringmol and GraphMol both represent organisms as strings of computer code that react with each other to produce new strings. One of the things these strings can do is copy each other, by executing an appropriate sequence of instructions from their language.

Stringmol

Stringmol has been used to implement an embodied copy operation. In terms of algorithm 1, it has embodied start and at-end functions, a stochastic char-copy function, and a crisp next function.

Stringmol's embodied copying process has been shown to produce interesting emergent behaviour (Hickinbotham et al., 2010a). The copying phenomenon is embodied in a *replicase* string that copies other instances of itself that it finds in the world. Stochastic char-copy errors pro-

duce slightly different versions of the replicase molecule that embody the replicase process; the altered replicase functionality potentially then generates very different molecules (species). One sequence of changes observed in Stringmol, described in detail in (Hickinbotham et al., 2010a), led to the system developing two different types of novel behaviour:

The only type of mutation programmed into Stringmol is a stochastic char-copy function, that can make single-character substitutions. Because of its embodiment, the copying phenomenon was able to evolve a different type of mutation: a macro-mutation that removes the first few characters. This is meta-evolution: the evolution of one of the mechanisms of evolution (the mutation function). And this meta-evolution happened because of embodiment: the embodied start and at-end functions could be exploited by the system in a novel way.

The copying mechanism initially programmed into Stringmol is one chemical copying another instance of itself:

$$A + A \mapsto A + A + A \tag{1}$$

By using the evolved macro mutation, along with other details of Stringmol's embodiment, the system produced a copying mechanism where two different strings that can no longer self-copy nevertheless self-sustain by copying each other (a hypercycle):

$$A + B \mapsto A + B + A \tag{2}$$

$$B + A \mapsto A + B + B \tag{3}$$

Figure 6 shows the embodiment structure of Stringmol. Embodiment allows the intermediate stages of the copying

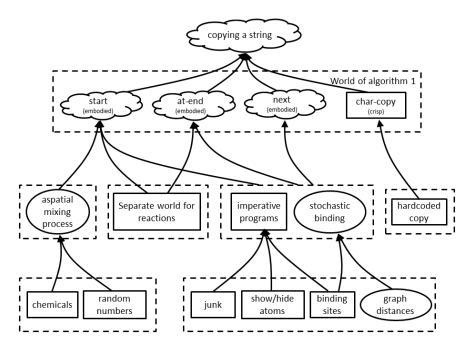


Figure 7: The mechanisms and worlds used by GraphMol to embody the phenomenon of copying a string.

process to be exploited and changed. This shows the potential power of embodying the copying process, or more generally, any process.

GraphMol

Graphmol is another related implementation of an embodied copy operation. In terms of algorithm 1, it has embodied start, at-end and next functions, and a crisp char-copy function.

Because of its embodied next function and crisp charcopy function, GraphMol's copy operation produces no single-character substitutions. Instead, all of its variation comes from the next function advancing down the string too quickly (causing small deletions in the copied string) or too slowly (causing small insertions in the copied string). There is a further combination of these two effects, where large insertions can occur. For more details of these mechanisms, see (Nellis, 2012, ch.6).

GraphMol uses a *replicase* similar to Stringmol, that self-sustains by copying other instances of itself. But Graph-Mol's population dynamics are very different from Stringmol's. In Stringmol, there is a single dominant *species* composed of identical copies of the same individual. Mutant individuals are created by the copying process; these mutants either die out very quickly or displace the incumbent species in the world (with the exception of the pairwise copying example above). In GraphMol, we see the initial replicase diversifying into a quasispecies of closely-related mutants that all copy each other. Each individual member of the quasispecies exists in very small numbers, and the composition of

the quasispecies changes continually over time, but the number of individuals making up the quasispecies stays constant (with stochastic fluctuations). This stable quasispecies lives for a time, before being killed by an unstable quasispecies of parasite mutants.

Comparisons via embodiment

Figures 6 and 7 show the embodiment structure of Stringmol and GraphMol in terms of our framework. These figures highlight that the novelty-generation capabilities of a system stem from the system's embodiment, which is a hierarchical nesting of different worlds. Different embodiments of the same phenomenon lead to different hierarchies of worlds.

Stringmol's novelty-generation comes mostly from its embodied start and at-end functions. Individuals are embodied in the Stringmol world as strings, interacting with each other via a 'soft' binding process. GraphMol's novelty-generation comes mostly from its embodied next function. Individuals are embodied in the GraphMol world as graphs that bind and run programs based on graph distances. This is a computational version of how biological machines operate on each other.

In both cases, we are in the domain of artificial chemistries. The domain informs the computational model, which defines the different *worlds* that need to be implemented to produce a running computer program. (The worlds are the enclosing rectangles in figures 6 and 7.) The influence of the domain, reflected in these worlds, filters up through the layers of embodiment. The choices made when designing the domain manifest as the novelty-generation

that we observe in the phenomenon of interest (the cloud at the top of the embodiment hierarchy).

Different properties of novelty-generation systems can be seen in the trees representing their embodiment hierarchies:

- In parts of the embodiment hierarchy that are more embodied (have deeper trees), we see greater noveltygeneration displayed in the running system (Stringmol's start and at-end function; GraphMol's next function).
- In parts that are less embodied, we do not see these aspects of the system playing key roles in the system's examples of novelty-generation (Stringmol's char-copy and next functions; GraphMol's char-copy function).
- 3. In parts that are of mixed depth, we see some influence on novelty-generation, but not as much as that shown by the more embodied parts (GraphMol's start and at-end functions are embodied, but are not as influential as its next function).

Conclusions

We have described our computational model of noveltygeneration, which involves implementing a model of evolution as an emergent property of virtual organisms living and interacting in a virtual world.

A key property of these virtual systems is embodiment: being able to embody phenomena of interest within a rich virtual world. We have given our definition of embodiment in terms of *phenomena*, *mechanisms*, *and worlds*. We have used our definition to give a description of Stringmol and GraphMol, each designed for computational novelty generation. Because of their different embodiments, Stringmol and GraphMol operate in different ways and produce different results. Stringmol exhibits macro-mutation and two-chemical copying; GraphMol exhibits two types of quasispecies, cooperative and parasitic. These two systems use the same domain (emergent evolution) and metamodel (machines copying strings), but different computational models.

Our definition of embodiment in terms of hierarchical phenomena, mechanisms and worlds allows us to do two different things. Firstly, given a metamodel (an abstraction of a domain of interest), it allows us to produce a space of different computational models that we can choose between. Secondly, given a computational model (design for a computer program), it allows us to characterise and diagram the model, allowing it to be easily compared with other models.

Building embodiment into our computational models allows us to start tackling the huge challenge of computational novelty generation. The descriptions of two example systems, Stringmol and GraphMol, provide an initial look into this space and a starting point to build on. The idea of embodiment in terms of *phenomena*, *mechanisms*, *and worlds* provides a conceptual framework for navigating this space.

Acknowledgements

Our thanks to Simon Hickinbotham for detailed comments on an earlier draft of this paper. This work was funded by the Plazzmid project, EPSRC grant EP/F031033/1.

References

- Cramer, N. L. (1985). A Representation for the Adaptive Generation of Simple Sequential Programs. In *Proc. 1st International Conf. on Genetic Algorithms*, pages 183–187. L. Erlbaum Associates Inc.
- Darwin, C. (1859). On the origin of species by means of natural selection. John Murray, 1st edition.
- Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley.
- Hickinbotham, S., Clark, E., Stepney, S., Clarke, T., Nellis, A., Pay, M., and Young, P. (2010a). Diversity from a monoculture: Effects of mutation-on-copy in a string-based artificial chemistry. In *ALife XII*, pages 24–31. MIT Press.
- Hickinbotham, S., Clark, E., Stepney, S., Clarke, T., Nellis, A., Pay, M., and Young, P. (2010b). Specification of the stringmol chemical programming language version 0.1. Technical Report YCS-2010-457, University of York.
- Hickinbotham, S., Clark, E., Stepney, S., Clarke, T., Nellis, A., Pay, M., and Young, P. (2011). Molecular microprograms. In ECAL 2009, Budapest, Hungary, volume 5777 of LNCS, pages 297–304. Springer.
- Holland, J. H. (1992). Adaptation in Natural and Artificial Systems. MIT Press.
- Kauffman, S. A. (2000). Investigations. Oxford University Press.
- Koza, J. R. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press.
- Lovelace, A. (1843). Note G. In Taylor, R., editor, Scientific Memoirs: Selected from the Transactions of Foreign Academies of Science and Learned Societies, and from Foreign Journals, volume 3, page 722.
- Maturana, H. and Varela, F. (1980). Autopoiesis and Cognition: the Realization of the Living. D. Reidel.
- Nellis, A. (2012). *Towards meta-evolution via embodiment in artificial chemistries*. PhD thesis, University of York.
- Nellis, A. and Stepney, S. (2011). Embodied copying for richer evolution. In *ECAL 2011*, pages 597–604. MIT Press.
- Quick, T., Dautenhahn, K., Nehaniv, C. L., and Roberts, G. (2000). The essence of embodiment. AIP Conference Proceedings, 517(1):649–660.
- Stepney, S. (2007). Embodiment. In Flower, D. and Timmis, J., editors, *In Silico Immunology*, pages 265–288. Springer.
- Varela, F. J. (1991). Organism: A meshwork of selfless selves. In Tauber, A. I., editor, *Organism and the Origins of Self*, pages 79–107. Springer.

Quantifying robustness in a spatial model of metabolism-boundary co-construction

Eran Agmon^{1,2}, Alexander J. Gates^{2,1}, Valentin Churavy³, & Randall D. Beer^{1,2}

¹ Cognitive Science Program, Indiana University, Bloomington, IN 47406, USA
 ² School of Informatics and Computing, Indiana University, Bloomington, IN 47406, USA
 ³ Institute of Cognitive Science, AG Neurocybernetics, Osnabrueck University, 49076 Osnabrueck, Germany

Abstract

We introduce a spatial model of autopoiesis that demonstrates boundary-metabolism co-construction, disintegration following harsh perturbations, and self-repair in response to lighter perturbations. The system's continued maintenance is equated with robustness and quantitatively investigated. This investigation highlights several important operational dependencies for robustness including spatiality, identity, stability, and appropriate specification of perturbations in the types of environments to which an entity must be robust.

Introduction

Autopoiesis is a proposed formal property of living systems, defined as "a network of processes of production (synthesis and destruction) of components, such that these components: 1) continuously regenerate and realize the network of processes that produces them, and 2) constitute the system as a distinguishable unity in the domain in which they exist" (Weber and Varela, 2002). This provides a unique characterization of life, which shifts the focus from the material properties of living systems to their systemic organization. Living systems retain an autopoietic organization despite material and energetic exchange with the environment, but with death this organization is lost and the system disintegrates into its respective components. A key attribute of such autopoietic systems is their robustness, or their ability to maintain their organization and repair their structure following environmental perturbations.

Elaborations on this concept in the field of enaction have argued that behaviors instantiated by autopoietic systems, which help sustain the basic organization, are equivalent to the processes of cognition. This leads to a radical reconceptualization of cognition, claiming that the inherent precariousness of autopoietic systems provides an intrinsic source of *normativity* (Weber and Varela, 2002) and an autopoietic system's ability to evade its impending disintegration is the foundation for *adaptivity* (Di Paolo, 2005).

However, the original criteria of autopoiesis are highly abstract, and their consequences cannot be verified until we understand the more basic premises. Minimal computational models have been used to ground the study of autopoiesis by observing the emergence of self-constructing systems out of a simulated environment, and to demonstrate the autopoietic logic through the analysis of these systems. Several models of autopoiesis have been developed over the last 40 years (McMullin, 2004). Different modeling approaches highlight different aspects of autopoiesis, but all of them share the assumption that abstractions of molecular interactions are the appropriate setting for studying autopoiesis.

In this paper, we put forth a lattice-based model of concentration dynamics. We demonstrate several properties of autopoiesis that arise in this model, including the roles of containment and metabolism, the precariousness of these systems, and their capacity for self-repair. We then transition to a systematic study of the system's robustness. We propose robustness as a fundamental property of successful autopoietic systems; they survive contact with the world despite their vulnerability to disintegration. We implement several types of perturbations on the virtual autopoietic systems, and through systematic study we observe how the space of possible perturbations can lead to these system's disintegration, robust self-repair, or transformation to new configurations.

Computational Autopoiesis

Current models of autopoiesis can be divided into four broad classes: the first type accounts for reaction kinetics but not spatiality, the second accounts for basic spatiality but not boundary construction, the third has an explicitly-defined boundary, and the fourth class consists of spatial models with distinct constructed boundaries.

We begin with arguably the most minimal type of model, which emphasizes reaction kinetics and ignores the spatial dimensions of molecular systems. This is seen in the hypercycle (Eigen, 1978), autocatalysis (Kauffman, 1986), chemical organization theory (Dittrich and Di Fenizio, 2007), and lambda calculus models (Fontana and Buss, 1994). These models implicitly assume homogeneity, such that all reactants interact equally with all other reactants. Reaction-based models are often more interested in the catalytic closure of a given chemical organization, which emphasizes

how molecular sets can synthesize themselves through the reactions of their molecular components. A type of precariousness (the capacity to fall apart) and robustness has been suggested by (Barandiaran and Egbert, 2014) in a nonspatial model composed of ordinary differential equations. For this class of models, death is simply a stable attractor at which all relevant molecular concentrations are zero, and viable spaces are those that have a stable equilibrium at nonnegative values of the relevant molecular concentrations.

The second type of autopoietic model incorporates space but does not demonstrate the construction and maintenance of a boundary. For example, Virgo (2011) argues that spots seen in the Gray-Scott reaction diffusion system have the property of individuation. Gray-Scott spots are spatial patterns, seen in a narrow band of system parameter values. They demonstrate how spatiality allows stable patterns to arise through reactions that occur at different rates within a heterogeneous system. However, these spots have no distinct spatial boundary, making it difficult to argue that the spots define themselves as distinct unities in space, and that they are self-regulating with regards to a medium. Similarly, gliders in the game of life are members of this second class because they construct themselves as spatial entities without explicitly different boundary components (Beer, 2004).

The third type of model includes spatial boundaries that have their own unique physics explicitly defined in the model (Bourgine and Stewart, 2004). As an example, the model presented by Egbert and Di Paolo (2009) has a membrane that is predefined as a circle of connected springs. The sections of this boundary can grow or shrink based on the local number of membrane molecules. This demonstrates boundary-metabolism co-regulation: the boundary contains the require metabolic molecules and keeps them from spilling into the environment, and the boundary requires metabolism because metabolism constructs the membrane molecules. However, the boundary is not something that arises naturally out of the lower-level physics.

The fourth type consists of spatial models that demonstrate constructed membranes. This includes the original particle-based model of autopoiesis (Varela et al., 1974), which defines a two-dimensional lattice, for which each position can be occupied by one particle at a time, or remain empty. Three types of molecules are included: substrates, catalysts, and links (membrane molecules). Through these molecules' reactions, a spatial boundary arises and contains the internal metabolism that continues to construct it. Similar models have demonstrated growth, change of shape, oscillation, and self-reproduction (Zeleny, 1977), the formation of flexible membranes, and movement (Breyer et al., 1998). The fourth class of models has been extended by modeling particle interactions in a continuous space, and by taking a macroscopic perspective on concentration dynamics rather than modeling the movement of individual molecules (Ono, 2001). The model presented in this paper falls into

this class of models.

There are many lessons we can learn from these past efforts to model autopoiesis. In this paper, we emphasize that, for a model to address how an autopoietic organization can distinguish itself from an environment, none of the properties on which that distinction is based should be hardcoded into the model. We suggest that most models of autopoiesis to date have various shortcomings with regards to this requirement. The first type of model, which focuses on stoichiometric relations but ignores spatiality, implicitly hardcodes the distinction between system and environment. The second type of model does not account for a spatial boundary, and so the spatial patterns being studied cannot regulate their own boundary conditions and therefore do not make themselves distinct. The third type of model hardcodes the membrane's properties into the model, and so it does not arise naturally from the underlying physics. The fourth type of model, if implemented carefully, is the only class that has the potential to demonstrate both catalytic closure and maintenance of an emergent boundary. In addition to careful implementation that avoids hardcoding desired organizational properties, resulting models should be systematically investigated to enhance our theoretical understanding of their underlying processes. Most of the described models were provided purely as proofs of concept for the autopoietic logic, and were not examined further.

Anisotropic Spatial Model

We consider a minimal autocatalytic model consisting of three parts: 1) a set of molecules \mathcal{M} and the corresponding set of constructive and destructive reactions, 2) spatial diffusion dynamics biased by repulsion, and 3) an anisotropic repulsion potential. These components are combined in a 2-dimensional lattice model derived from a previous model investigated by Ono (2001). We have simplified the reaction kinetics, moved it to a rectangular lattice, and implemented continuous orientation fields. In addition, our model is completely deterministic.

For each of the molecular types $m \in \mathcal{M}$ and at each point in space x_{ij} , the equations of motion take the form of a system of coupled differential equations:

$$\frac{\partial m(x_{ij})}{\partial t} = \mathbf{R}_m(x_{ij}, t) + \mathbf{D}_m^{\theta}(x_{ij}, t) + \mathbf{\Gamma}_m(x_{ij}, t) \quad (1)$$

where $R_m(x_{ij},t)$ indicates rate contributions due to the mass-action assumptions of our chemical reactions, $D_m^{\theta}(x_{ij},t)$ indicates a diffusive term biased by repulsion and orientation field θ , and $\Gamma_m(x_{ij},t)$ denotes potential external contributions to the system. Each of these terms will be addressed below.

Autocatalytic stoichiometry. Our model captures the concentration dynamics of four distinct molecular types $\mathcal{M} = \{A, M, F, W\}$: the autocatalyst A, the membrane M,

the food F, and water W. Each of these molecular types reacts as follows. Autocatalyst A catalyses the replication of additional A and consumes F in the process, with a rate k_{AA} . Autocatalyst A also catalyzes membrane M from F at a rate k_{AM} . Both A and M decay at rates k_A and k_M respectively. Water W is an inert substance. These reactions are summarized by the following stoichiometric equations:

$$2A + F \xrightarrow{k_{AA}} 3A$$
 $2A + F \xrightarrow{k_{AM}} 2A + M$ (2)
 $A \xrightarrow{k_A} \emptyset$ $M \xrightarrow{k_M} \emptyset$

with all reverse reaction rates set to 0. Conservation of mass can be assumed via the introduction of an inert waste product which is instantaneously removed from the system. Finally, instead of modeling each individual molecule, we course-grain over space and consider positive concentrations of molecules. The mass-action assumptions allow us to convert these stoichiometric relations into dynamical equations:

$$\mathbf{R}_{A}(x_{ij},t) = k_{AA}F(x_{ij},t)A(x_{ij},t)^{2} - k_{A}A(x_{ij},t)$$

$$\mathbf{R}_{M}(x_{ij},t) = k_{AM}F(x_{ij},t)A(x_{ij},t)^{2} - k_{M}M(x_{ij},t)$$

$$\mathbf{R}_{F}(x_{ij},t) = -(k_{AA} + k_{AM})F(x_{ij},t)A(x_{ij},t)^{2}$$

$$\mathbf{R}_{W}(x_{ij},t) = 0$$
(3)

Unlike the aforementioned Ono (2001) model, the reaction rates are all constant.

Diffusion and repulsion. The above reactions take place on a $N_x \times N_y$ lattice. Each molecular concentration $m \in \mathcal{M}$ follows diffusive dynamics resulting in conserved concentration flows between the lattice sites:

$$\boldsymbol{D}_{m}^{\theta}(x_{ij},t) = d_{m} \sum_{x_{\alpha\beta} \in \mathcal{N}_{ij}} m(x_{\alpha\beta}) g_{m}^{\theta}(x_{\alpha\beta} \to x_{ij},t) \quad (4)$$
$$- m(x_{ij}) g_{m}^{\theta}(x_{ij} \to x_{\alpha\beta},t)$$

where d_m is the diffusion constant for the m-th type molecular concentration, \mathcal{N}_{ij} is the Moore neighborhood of each lattice site x_{ij} , and $g_m^\theta(x_{ij} \to x_{\alpha\beta})$ is the diffusive bias of concentration m from site x_{ij} to site $x_{\alpha\beta}$. This diffusive bias is given as a function of the gradient of a repulsion potential P_m^θ between the sites:

$$g_m^{\theta}(x_{ij} \to x_{\alpha\beta}, t) = \frac{\Delta P_m^{\theta}(x_{ij} \to x_{\alpha\beta}, t)}{1 - e^{-\Delta P_m^{\theta}(x_{ij} \to x_{\alpha\beta}, t)}}$$
 (5)

where $\Delta P_m^{\theta}(x_{ij} \to x_{\alpha\beta},t) = P_m^{\theta}(x_{ij},t) - P_m^{\theta}(x_{\alpha\beta},t)$ is the gradient of the repulsion potential P_m^{θ} in the direction of $x_{\alpha\beta}$. Finally, the repulsion potential $P_m^{\theta}(x_{ij},t)$ for molecular concentration type m at lattice site x_{ij} takes the form:

$$P_m^{\theta}(x_{ij}, t) = \sum_{\mu \in \mathcal{M}} \sum_{x_{\alpha\beta} \in \mathcal{N}_{ij}^*} \mu(x_{ij}, t) f_{m\mu}^{\theta}(x_{\alpha\beta}, x_{ij}) \quad (6)$$

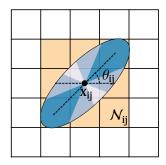


Figure 1: Anisotropic repulsive interaction ellipse. The ellipse is centered at lattice site x_{ij} with the major axis orientated at angle θ_{ij} with respect to the horizontal. Each of the 8 lattice sites in the Moore neighborhood \mathcal{N}_{ij} of x_{ij} (orange) is subject to an interaction proportional to the normalized area of the corresponding $\pi/4$ octant of the ellipse (blues). Darker shades of blue signify stronger repulsion.

which has contributions from all μ -th type concentrations in the inclusive Moore neighborhood \mathcal{N}_{ij}^* of x_{ij} . Within this framework, the specific interactions between concentration types m and μ are determined by the interaction strength $f_{m\mu}^{\theta}(x_{\alpha\beta},x_{ij})$. In general, these interactions can have different values depending on the relative location within the neighborhood (including self interactions). In the next section, we will explain how those interactions involving M are dependent on the orientation field θ .

Anisotropic membrane potentials. With isotropic interaction potentials, the membrane molecules behave like oil in water, forming large symmetric clumps without any containment properties, thus we turn to anisotropic interaction potentials to generate a boundary layer. Motivated by the dynamics of lipid sheets that form in water, we focus on an important property of boundaries; they are lower dimensional surfaces embedded in a higher-dimensional space. A natural consequence of this embedding is the introduction of an orientation field that describes the local orientation $\theta(x_{ij},t) \in [0,\pi)$ of the membrane concentration at each lattice site x_{ij} . This specifies the orientation of the major axis for an elliptical repulsion potential between the membrane concentration M and any other μ -th type concentration as shown in Figure 1. The interaction strength $f_{M,\mu}^{\theta}(x_{\alpha,\beta},x_{ij})=f_{M,\mu}(x_{\alpha\beta})C^{\theta}(x_{\alpha,\beta}-x_{ij})$ in each of the 8 neighboring cells $x_{\alpha\beta}\in\mathcal{N}_{ij}$ is calculated as the normalized area $C^{\theta}(x_{\alpha,\beta}-x_{ij})$ of the directionally appropriate $\pi/4$ octant of the ellipse, with the $-\pi/8$ to $\pi/8$ octant directed towards the central right neighbor $x_{i+1,j}$. The lengths of the major and minor axes of this ellipse are given by r_M^{major} and $r_M^{\rm minor}$ respectively.

Further mimicking the behavior of lipid sheets, the orientation of membrane concentrations $\theta(x_{ij}, t)$ at lattice site

 x_{ij} aligns with their neighbors with a force proportional to the concentration of membrane $M(x_{\alpha\beta},t)$ in the neighboring cell $x_{\alpha\beta}$. Thus, the local orientation has the following dynamics:

$$\frac{\partial \theta(x_{ij}, t)}{\partial t} = k_{\theta} \sum_{x_{\alpha\beta} \in \mathcal{N}_{ij}} M(x_{\alpha\beta}, t) \psi \left(\theta(x_{\alpha\beta}, t) - \theta(x_{ij}, t) \right)$$
(7)

where $\psi(x)=\sin(2x)$ was chosen so as to stabilize aligned orientations and destabilize anti-aligned ones in a smooth manner.

Full model. The full model consists of four differential equations in the form of equation (1) (one for each molecular concentration type $m \in \mathcal{M}$) and one differential equation for the orientation in the form of equation (7) for each site $x_{ij} \in N_x \times N_y$ the lattice. Thus, the model is technically a $5N_xN_y$ dimensional coupled, nonlinear dynamical system. Since many of the phenomena we are interested in occur in non-equilibrium, the $\Gamma_m(x_{ij},t)$ term allows for general flows of concentrations in or out of the system. This will be particularly useful for modifying the food concentration Fwhich plays the role of an energy supply for the chemical reactions, but does not participate in repulsive interactions. Additionally, we note that while the water concentration Wis reactively inert, it contributes to repulsive interactions and fills what would otherwise be empty space. The spatial distributions of the autocatalyst A and membrane M concentrations are of primary interest to the following analysis.

Model parameters. The general model supports a plethora of possible behaviors for the spatial distribution of the molecular concentrations. Although a systematic study of these parameters was not performed, our parameter selection was informed by two criteria; repulsion should allow membranes to form stable sheets, and the tradeoff between diffusion and production rates should support the requirement of containment for ongoing catalysis. The following parameter values were used; $k_{AA} = 0.09$, $k_{AM} = 0.1$, $k_A = k_M = 0.001, k_\theta = 0.0125, d_A = d_M = d_F =$ $d_W = 0.29$. For the repulsion interactions, we assume all interactions are symmetric with respect to molecular type (i.e. $f_{m\mu}^{\theta} = f_{\mu m}^{\theta}$), and take values: $f_{A,A}^{\theta}(|x_{\alpha,\beta} - x_{ij}| = 1) = 0.1875$, $f_{M,M}^{\theta}(|x_{\alpha,\beta} - x_{ij}| = 1) = 1.5 * C^{\theta}(x_{\alpha,\beta} - x_{ij})$ $(x_{ij}), f_{M,A}^{\theta}(|x_{\alpha,\beta} - x_{ij}| = 1) = f_{M,W}^{\theta}(|x_{\alpha,\beta} - x_{ij}| = 1)$ 1) = $7.0 * C^{\theta}(x_{\alpha,\beta} - x_{ij})$, where $C^{\theta}(x_{\alpha,\beta} - x_{ij})$ is the orientation-dependent ellipsoidal area with radii $r_M^{\rm major}=20$ and $r_M^{\rm minor}=1$. All other parameters were set to 0

The system we consider unfolds on a $N_x = N_y = 50$ square lattice with toroidal boundary conditions, making a total of 12,500 ordinary differential equations. To keep a steady supply of food entering the system, the concentration of F is driven towards a saturation value of s_F at a rate k_F via the additional linear term $\Gamma_F(x_{ij},t) = k_F(s_F - t_F)$

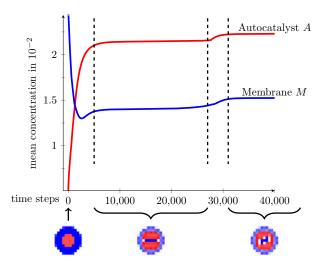


Figure 2: Autocatalyst and membrane configurations over a period of 40,000 time steps. These illustrate a short transient from an initial configuration, followed by a long stable configuration *SC*, and then a transition to a second stable configuration.

 $F(x_{ij},t)$), where we set $k_F=0.8$ and $s_F=0.18$. This only applied outside of a circle of radius 9 cell units. This area of food resupply does not overlap with the self-constructing configurations that we will explore, food can only reach the central area surrounding these configurations by diffusion. No additional interactions with an external source are assumed, thus $\Gamma_A(x_{ij},t)=\Gamma_M(x_{ij},t)=\Gamma_W(x_{ij},t)=0$.

Model Behavior

Initiation and time evolution. Given any initial spatial configuration for the molecular concentrations and orientation field, the system unfolds deterministically, most often towards a uniform state in which A and M concentrations have been depleted. To facilitate finding a viable stable configuration, with non-zero values of A and M, the system was initialized in the convenient configuration shown in Figure 2 at t = 0. This configuration consists of a small circle of autocatalyst with a diameter of five lattice cells across and a uniform concentration of 0.6. The circle of autocatalyst is surrounded by a ring of membrane that is 3 lattice cells thick and has a concentration of 0.8. The membrane orientation field is initiated as concentric circles around the lattice's center. Food is initiated as a uniform field with a concentration of 0.18, and water fills the environment to bring the summed concentrations of all molecules at each lattice cell to a value of 1.0. This cell-like initial configuration, with autocatalyst contained by a membrane boundary, facilitates the subsequent unfolding towards a viable configuration. No systematic study of initial configurations was performed.

We are primarily interested in the first stable configura-

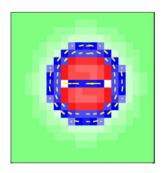


Figure 3: Stable configuration SC, showing the central 19×19 cells of the lattice. SC is composed of autocatalyst (red) and membrane (blue), and is embedded in an environment that is refilled with food (green). This food field is omitted in all other figures. The membrane's orientation is shown by the yellow lines. Brighter regions designate reduced molecular concentrations.

tion (shown in greater detail in Figure 3), which arises from the initial configuration after a short transient, and then persists practically unaltered from t=5000 to t=27000. This timescale is much longer than the timescale of typical cell processes and death events, so we treat the structure as stable given the processes of interest. The configuration of the system at t=10000 will be resimulated and tested for the remainder of this paper. This configuration will be called the stable configuration, or SC. In SC, the membrane is thinner than the initial configuration, and there is an interesting bridge formation of membrane cutting across the center of the cell and splitting the autocatalyst into two regions. Cells with low molecular concentration appear directly to the left and the right side of the bridge, this is due to the directional repulsion of the anisotropic membrane molecules.

SC eventually destabilizes due to small changes in concentrations that accumulate over time. The destabilized structure goes through a short transient before restabilizing at a different configuration.

Containment and metabolism. Of particular importance to models of autopoiesis is the establishment of metabolism-boundary co-construction. While the boundary's dependence on metabolism is explicitly made through the reactions in our model, the other direction needs a bit more justification. Figure 4 demonstrates the role of containment and metabolism by pulling apart the membrane and autocatalyst fields of SC. The membrane field is displaced upwards by 25 lattice cells from its original position and the membrane orientation is copied. Everything else is kept equal. Whereas the stable configuration would have lasted for approximately 25000 time steps, the displaced configuration is clearly disintegrating after 500 time steps, and the final vestiges of the relevant molecules disintegrate around t=3000.

These results illustrate the strong symmetrical depen-

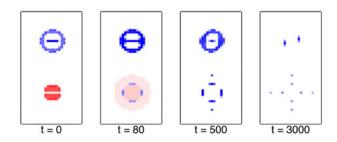


Figure 4: The co-dependent roles of containment and metabolism are illustrated by separating membrane from autocatalyst, and showing the system's rapid disintegration.

dence of boundary and metabolism in the processes of coconstruction. The processes of containment and molecular synthesis counter the effects of diffusion and decay. The autocatalyst needs to be contained by the membrane in order to remain at the high concentrations that allow for its reactions of production. Without the membrane, autocatalyst diffuses away and quickly decays. This dependence is shown in 4; as the autocatalyst diffuses, it begins to construct membrane (shown as thin blue lines surrounding the autocatalyst starting at t = 80), but the rate of construction is not sufficient to contain the autocatalyst before its complete decay. This figure also shows that the boundary needs the autocatalyst for its synthesis. The boundary's shape begins to change immediately as it thickens where the autocatalyst once was, and begins to decay. Within several hundred time steps, only vestiges of the boundary survive and soon after they are gone. Only through coupled co-constructive processes can either molecular species survive.

Self-repair and disintegration. Figure 5 illustrates SC's precariousness and its capacity for self-repair. In the figure, two sequences are shown in which a tear perturbation was applied to the membrane of SC, with a larger tear in the bottom sequence. A tear is a perturbation in which a small section of membrane is removed by setting all the concentrations to a value of 0, and randomizing the orientation of those lattice cells. At t=1, we can see the missing membrane section in both sequences.

In the top sequence, which has a smaller tear, the membrane flows back into the removed section and the system fully restabilizes by t=500. The orientation, which is not illustrated in the figure, also realigns and allows the membrane concentration to flow around the boundary. Interestingly, this event results in a different stable configuration than before the perturbation. The cell is elongated on the top side where it suffered the perturbation, resembling residual outgrowth.

In the bottom sequence, with the larger tear, the autocatalyst begins to leak out, as shown at t=15. Membrane also flows into the tear, but it is repulsed by the autocatalyst and

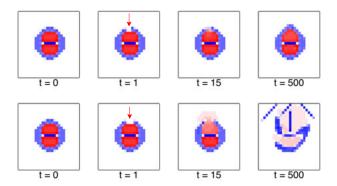


Figure 5: *SC*'s precariousness and capacity for self-repair are illustrated by applying two different-sized tears, pointed to with the red arrows. The upper sequence, which sustains a smaller tear, is able to repair itself. The lower sequence, with the larger tear, ends in disintegration.

cannot repair the hole. By t=500, most of the autocatalyst has leaked out, and the membrane is in a distorted form. The leaking autocatalyst creates new membrane molecules, which self-assemble additional strings in the environment. With the containment condition broken, the metabolism cannot persist and the boundary decays. Most of the structure has been depleted by t=1000, and soon after the system returns to its terminal, uniform state.

Perturbation Studies

In the behavioral studies of the previous section, we observed conditions that allow the membrane and autocatalyst to sustain each other through co-construction, and repair themselves following tear perturbations. We now shift to a more quantitative approach for investigating the limits of the system's viability and its robustness to perturbation. Here, robustness is simply defined as the capacity to retain configurational stability despite harmful perturbations. We study these properties by systematically applying perturbations and observing which conditions drive the system to disintegration, and which allow it to remain viable.

Quantifying robustness immediately highlights three challenges: 1) How to quantify viable stability, 2) How to characterize a system perturbation, and 3) How to characterize a system's response to a perturbation.

Given that the death state is a stable attractor for which the relevant concentrations are depleted, viability exists in stable, nonzero configurations of autocatalyst and membrane. We associate stability with extended periods of time in which all of A and M's time derivatives are near zero. Thus, a structural configuration is stable when the sum of the temporal derivatives of A and M over the whole system satisfies $\sum_{x_{ij}} |\dot{A}(x_{ij},t_k)| < \epsilon$ and $\sum_{x_{ij}} |\dot{M}(x_{ij},t_k)| < \epsilon$ for all $t_k \in [t,t+5000]$, and a sufficiently small $\epsilon=0.01$.

There are many types of system perturbations that could

be systematically explored. We classify the space of possible perturbations by three main dimensions: type, location, and time. Type refers to the five different state variables: the molecular types (A, M, F, and W) and the orientation field of the membrane (θ) . Perturbations can be applied to any combination of these variables. Location refers to the positions within the lattice at which these molecules are perturbed. Perturbations can be applied to spatially localized regions, or to all points throughout the system. Time refers to the temporal structure of perturbations throughout a simulation.

Ideally, each class of perturbations will have a few parameters which capture the perturbation and allow one to explore the system's robustness as the parameters are varied. The tear perturbation is an instantaneous, spatially-local perturbation, applied to M and θ and parameterized by the spatial extent of the tear. On the other hand, global noise is an instantaneous perturbation that modifies every variable in the system. The space of possible perturbations is vast.

Finally, we must characterize a system's response to a perturbation. Here, we measure how long it takes for a system to reach stability following a perturbation. When it stabilizes, we determine its structural distance from the original configuration by taking the summed absolute difference between the two configuration's states. If this sum is near zero, then the system has restabilized at the original configuration. In contrast, larger values indicate a new configuration has been reached.

Gaussian blur perturbations. The Gaussian blur is a global perturbation applied instantaneously to the four molecular species (but not the orientation field θ). Gaussian blur is a common technique from image processing used to make an image blurry or reduce image noise. For our model, applying the blurring function is akin to instantaneously shaking the system. More shaking leads to more blurring, as molecules between local regions are mixed. Mathematically, the Gaussian blur is a convolution of the concentration fields with a Gaussian function, resulting in a Gaussian-weighted spatial average. By increasing the variance (σ^2) of the Gaussian function, we can make the blurring increasingly severe.

Figure 6 shows the time to restabilization following a Gaussian blur perturbation as σ^2 is systematically varied. In all instances the system was initialized to SC. The figure shows that the restabilization time varies as the blur becomes more severe until, for $\sigma^2 > 4.9$, the system becomes too scrambled and can no longer recover and instead evolves to the stable terminal attractor.

Figure 6 also illustrates four different stable viable configurations that are reached following perturbations. Configuration A, which we observe at small σ^2 , is SC. As σ^2 is increased it takes slightly longer to restabilize at A. At $\sigma^2 > 3.4$ we see a discrete change occur, in which the system takes much longer to stabilize, and when it stabilizes it

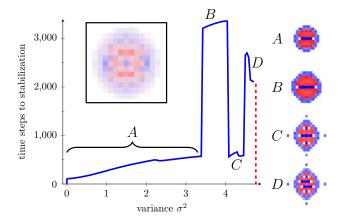


Figure 6: Gaussian blur perturbation to SC. An example blur at $\sigma^2=4$ is shown in the inset. As σ^2 is increased from 0 it takes longer to stabilize back to the original configuration. At $\sigma^2>3.4$ the blur perturbation drives the system to different stable configurations $(B,\,C,\,$ and D). A $\sigma^2>4.9,\,$ marked with the dashed red line, leads to the system's disintegration.

is at a new configuration B. This configuration looks similar to A, but has a diameter 2 cells larger. As we increase σ^2 beyond 4.1, configuration C is reached, and at 4.5 we see configuration D. For configurations C and D, we observe small membrane structures that have formed external to the boundary, there are also internal differences from A and B. The external points are stable, and are sustained by a slow and constant diffusion of autocatalyst that spreads outside of the boundary. In D, we observe a broken symmetry in the internal membrane stripes. Presumably there is another stable configuration identical to this, but with reversed broken symmetry.

Spatially-localized Gaussian perturbations. Finally, we explore two variants of a spatially-localized Gaussian perturbation. These are applied in space by using a translated Gaussian function to multiply concentrations of the four molecular species (A, M, F, and W) in the region around a given focal point. The function is of the form:

$$G(x_{ij}) = \alpha e^{-(|x_{ij} - x_f|)^2 / 2\sigma^2} + 1$$
 (8)

where $|x_{ij}-x_f|$ is the distance from the focal point x_f and the given cell x_{ij} , $\alpha \in [-1,\infty)$ is the amplitude of the function, and σ^2 is the variance. An α of 0 produces no change to the fields because the focal point and the points around it are multiplied by 1. Negative α reduces the field concentration while positive α increases the concentrations of the affected cells surrounding the focal point.

Figure 7 shows this class of perturbations applied to two different focal points of SC. The locations of the focal points

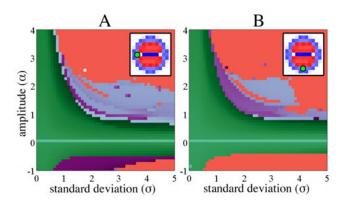


Figure 7: A localized Gaussian perturbation applied to two focal points (shown by the green dots on the inset diagrams) on the stable configuration *SC*. Perturbations where the system: (green) restabilized back to *SC*, (purple) stabilized at a different configuration, and (red) resulted in the system's disintegration. Darker shades indicate longer times to restabilization following perturbation.

are shown by the green dots on the inset configuration diagrams, and the results are shown in the two perturbation figures. In these plots, SC restabilization is shown in green, stabilization to a different configuration is shown in purple, and perturbations that lead to death are shown in red. Time to restabilization is shown by the shade, with darker shading indicates longer time to restabilization. This highlights several features. Since $\alpha=0$ perturbations have no effect on SC, the corresponding light green line indicates zero timesteps to regain stability. The shading also highlights that as a boundary between different configurations is approached, it takes increasingly longer to restabilize.

A qualitative comparison of the two perturbation figures reveals that the location of perturbation significantly alters the systems response. In Figure 7A, there is less red overall, meaning that fewer perturbations lead to SC's disintegration. In fact, we see that the red area from Figure 7B is mostly purple in 7A. For example, perturbations with low α and mid-range σ lead to disintegration when applied to the focal point in 7B, but when applied to the focal point in 7A, they merely drive the system to a different configuration.

Discussion

Robustness of boundary-metabolism co-construction is a key property of autopoietic systems. Living systems do not exist in perfect conditions, and are prone to environmental perturbation. Because of this, successful autopoietic systems must be able to repair themselves in order to persist. We outlined a minimal spatial model that exhibits metabolism-boundary co-construction and emergent organization. This allowed us to simulate stable, mutually

dependent, configurations of autocatalyst and membrane molecules, and to test their behavioral properties under environmental perturbations.

Spatiality adds an important dimension to our understanding of autopoiesis. Instead of imposing a system/environment distinction upon our model, an identity emerges as a sustained spatial inhomogeneity. Precariousness then becomes the potential for the network of interactions to explicitly dissolve. While we can qualitatively observe the system spatially disintegrate, we currently lack a quantitative measure of precariousness. Therefore, we used the uniform system state as an indicator that dissolution has occurred, rather than its definition. This contrasts with nonspatial models that define disintegration as relevant variables equaling zero (Barandiaran and Egbert, 2014), or as essential variables leaving explicitly-defined viability limits (Ashby, 1952).

Our quantitative analysis of robustness for boundarymetabolism co-construction highlighted several important operational dependencies for the concept. A formalization of robustness is highly dependent on the definition of perturbation to which the configuration must be robust. Here, we laid out a few key dimensions that describe the space of spatial-temporal perturbations and showed quantitatively distinct descriptions of robustness emerged as different perturbations were considered. Robustness is also dependent on the identity criteria used to establish organizational equivalence. For example, should the configuration with a small outgrowth resulting from the small tear perturbation in Figure 5 be considered the same organization as the original stable configuration SC? How about the equivalence between SC and the configurations with differing internal structure following the Gaussian blur in Figure 6B? Finally, our quantitative measure for robustness is dependent on the particular method for evaluating the stability of steady-state configurations. The shortcomings of this definition were particularly evident when we determined the original stable configuration SC eventually transforms into a qualitatively different stable configuration after a significantly long time.

One unexplored phenomenon observed in our model is the emergence of dynamic behavior. Behaviors can arise from heterogeneous environments or from the temporal dependence of a system's robustness profile. For example, food gradients might result in a form of mobility due to asymettries in the configuration's local environment (Egbert and Di Paolo, 2009). As a system is exposed to different perturbations its configuration can be altered, which would change its robustness profile and behavioral responses following future perturbations. Extensions of identity and stability that allow one to characterize spatial configurations engaging in dynamic behaviors are still lacking. Of particular interest are configurations that have a disposition for being perturbed towards increasingly robust configurations, which might relate to the notion of adaptivity in enaction (Di Paolo, 2005).

Computational models, like the one presented here, provide a vehicle for exploring such concepts.

Acknowledgments

We would like to thank Eduardo Izquierdo, Matthew Egbert, and three anonymous reviewers for feedback on an earlier draft of this paper. This work was supported in part by NSF grant IIS-0916409 to RDB, as well as NSF IGERT fellowships to EA and AJG. The software is available for download at https://github.com/vchuravy/spatial_self_construction.

References

- Ashby, W. R. (1952). Design for a brain. Wiley.
- Barandiaran, X. E. and Egbert, M. D. (2014). Norm-establishing and norm-following in autonomous agency. *Artificial Life*, 20(1):5–28.
- Beer, R. D. (2004). Autopoiesis and cognition in the game of life. *Artificial Life*, 10(3):309–326.
- Bourgine, P. and Stewart, J. (2004). Autopoiesis and cognition. *Artificial Life*, 10(3):327–345.
- Breyer, J., Ackermann, J., and McCaskill, J. (1998). Evolving reaction-diffusion ecosystems with self-assembling structures in thin films. *Artificial Life*, 4(1):25–40.
- Di Paolo, E. A. (2005). Autopoiesis, adaptivity, teleology, agency. *Phenomenology and the Cognitive Sciences*, 4(4):429–452.
- Dittrich, P. and Di Fenizio, P. S. (2007). Chemical organisation theory. *Bulletin of mathematical biology*, 69(4):1199–1231.
- Egbert, M. D. and Di Paolo, E. (2009). Integrating autopoiesis and behavior: An exploration in computational chemo-ethology. *Adaptive Behavior*, 17(5):387–401.
- Eigen, M. (1978). The hypercycle: A principle of natural self-organization. *International Journal of Quantum Chemistry*, 14(S5):219–219.
- Fontana, W. and Buss, L. W. (1994). the arrival of the fittest: Toward a theory of biological organization. *Bulletin of Mathematical Biology*, 56(1):1–64.
- Kauffman, S. A. (1986). Autocatalytic sets of proteins. *Journal of Theoretical Biology*, 119(1):1–24.
- McMullin, B. (2004). Thirty years of computational autopoiesis: A review. *Artificial Life*, 10(3):277–295.
- Ono, N. (2001). Artificial chemistry: Computational studies on the emergence of self-reproducing units. PhD thesis, Institute of Physics, The University of Tokyo.
- Varela, F. G., Maturana, H. R., and Uribe, R. (1974). Autopoiesis: the organization of living systems, its characterization and a model. *Biosystems*, 5(4):187–196.
- Virgo, N. D. (2011). *Thermodynamics and the structure of living systems*. PhD thesis, University of Sussex.
- Weber, A. and Varela, F. J. (2002). Life after Kant: Natural purposes and the autopoietic foundations of biological individuality. *Phenomenology and the Cognitive Sciences*, 1(2):97–125.
- Zeleny, M. (1977). Self-organization of living systems: A formal model of autopoiesis. *International Journal of General Systems*, 4(1):13–28.

Self-referencing cellular automata: A model of the evolution of information control in biological systems

Theodore P. Pavlic¹, Alyssa M. Adams¹, Paul C. W. Davies¹ and Sara Imari Walker¹

¹Arizona State University, Tempe, AZ 85287 tpavlic@asu.edu

Abstract

Cellular automata have been useful artificial models for exploring how relatively simple rules combined with spatial memory can give rise to complex emergent patterns. Moreover, studying the dynamics of how rules emerge under artificial selection for function has recently become a powerful tool for understanding how evolution can innovate within its genetic rule space. However, conventional cellular automata lack the kind of state feedback that is surely present in natural evolving systems. Each new generation of a population leaves an indelible mark on its environment and thus affects the selective pressures that shape future generations of that population. To model this phenomenon, we have augmented traditional cellular automata with state-dependent feedback. Rather than generating automata executions from an initial condition and a static rule, we introduce mappings which generate iteration rules from the cellular automaton itself. We show that these new automata contain disconnected regions which locally act like conventional automata, thus encapsulating multiple functions into one structure. Consequently, we have provided a new model for processes like cell differentiation. Finally, by studying the size of these regions, we provide additional evidence that the dynamics of self-reference may be critical to understanding the evolution of natural language. In particular, the rules of elementary cellular automata appear to be distributed in the same way as words in the corpus of a natural language.

Introduction

Cellular automata (CA) are model complex systems that combine spatial memory with relatively simple update rules to produce rich dynamic patterns (Wolfram, 2002). In this regard, they can be viewed as models for life. "Rules" in DNA encode policies that iteratively react to the environment by modifying it. Thus, over evolutionary time scales, natural selection can explore the nucleic-acid rule space and amplify those rules which provide useful functions. With this narrative in mind, researchers have used *in silico* artificial selection to explore the CA rule space and amplify rules with certain computational abilities (Breukelaar and Bäck, 2005; Das et al., 1994; Hordijk, 2013; Mitchell et al., 1994). Moreover, there has been much interest in understanding the demographic dynamics of these CA rule populations over

their evolutionary history (Hordijk, 2013; Mitchell et al., 1994). That is, artificial selection of these evolving CA's has itself become a model for the innovation intrinsic to natural selection.

One major difference between evolving cellular automata (EvCA) and evolving natural organisms is the lack of feedback in the fitness channel of the former. In EvCA, each new generation faces the same selective pressures as prior generations. However, with new generations of natural organisms, there is feedback between the current demographics and the selective pressures shaping future demographics. Goldenfeld and Woese (2011) point out that these self-referential dynamics are a unique characteristic of life – making life distinctly different from any other physical system. Two of us (SIW and PCWD) have proposed that self-referential dynamics are one of the hallmarks of life, emerging with its origin (Walker and Davies, 2013). Where EvCA's will only innovate in the presence of external "abiotic" pressures, natural organisms put pressure on themselves to re-organize even without an external fitness driver. Similarly, at everyday and ontogenetic time scales, conventional CA's do not easily embed the regulatory mechanisms that permeate throughout life. A single genome gives rise to a wide variety of differentiated cell phenotypes which, locally, appear to follow a consistent set of operational rules but globally appear to have no shared program. By modeling feedback explicitly, these phenomena can be explained using gene regulatory network (GRN) frameworks (Davidson, 2010; Schlitt and Brazma, 2007), where the expression level of one gene promotes or inhibits the expression level of another and thus "latches" cells into different types. Thus, by making feedback between state and dynamics explicit in CA's, it may be possible to enrich their ability to model how organisms emerge, evolve, develop, and react to both their external environment and their internal state.

Here, we introduce self-referencing cellular-automaton framework we call PICARD where *PICARD Implements CA Rules Differently* (PICARD). Like a traditional one-dimensional CA, PICARD executions move from one iteration to another by some rule. However, whereas traditional

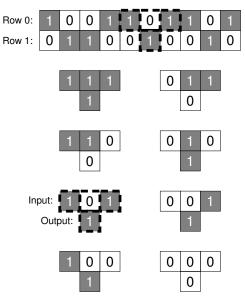
CA's require the rule to be static and externally specified, PICARD infers the iteration rule from the current state of the CA itself. As we will show, executions from multiple static CA's can be embedded within a single PICARD – a PICARD can be identical to one static CA from certain initial conditions and another static CA from other initial conditions. Thus, a PICARD can combine the computational abilities of different CA's within one entity. Moreover, whereas CA's differ by their rule, PICARD's differ by their state-to-rule mapping. Because there are many more state-to-rule mappings than there are CA rules, the PICARD parameter space is potentially much richer for later evolutionary investigations.

To some extent, PICARD is a simple attempt to add dvnamical feedback that is missing in traditional evolutionary cellular automata. However, because iterations are generated by rules that are encoded in previous iterations, PI-CARD feedback is the kind of self-reference that is thought to be a characteristic feature of life (Goldenfeld and Woese, 2011; Hofstadter, 1979; Kataoka and Kaneko, 2000a,b; Walker and Davies, 2013). Our CA approach shares many similarities with self-referencing functional-dynamics developed by Kataoka and Kaneko to model the evolution of rules (Kataoka and Kaneko, 2000a,b). Attempting to avoid the biochemical complexities of the evolution of nucleic acids, they turn their focus on the evolution of natural language. Furthermore, they draw connections between attractors in their coupled-logistic-map landscape and words that accumulate in language. Although their framework is very different than the automata we study here, we too have results that appear to be strongly connected to the evolution of natural language. Thus, augmenting cellular automata with self-reference widens our ability to model to evolution of language in unanticipated ways.

Cellular Automata with PICARD Mappings

Although PICARD implements CA rules differently, once each rule is defined, a PICARD iteration is identical to an iteration of a conventional elementary CA. As shown in Fig. 1(a), a traditional elementary CA generates each row based on the pattern in the preceding row. The iteration rule is a lookup table that maps each triplet of bits in the preceding row to a single bit in the following row. Thus, with the right initial conditions, some rules can produce intricate patterns over many generations, as shown in Fig. 1(b).

Where PICARD differs from a traditional CA is that no static rule is specified. Instead, a map is provided from each row to the rule that will operate on it. We are primarily interested in CA's with more than eight cells; consequently, this mapping is a coarse graining of the system – some rules will necessarily correspond to multiple different row patterns. With this multiple realizability of rules in mind, each PICARD row and rule can be viewed as a *microstate-macrostate* pair. Consequently, in the following, we will use





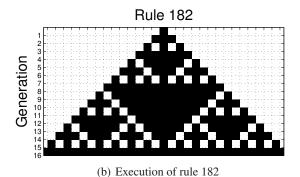


Figure 1: Traditional cellular automaton. In (a), a single iteration of a traditional CA is shown in the top two rows, where row 0 is taken as an initial condition on which a static CA rule operates to produce row 1. The static CA rule is summarized in the eight groups of four cells in the bottom of the figure. In each group, the top three cells match adjacent cells in row 0, and the bottom one cell represents the cell generated beneath the row-0 cells in row 1. One such grouping in the two rows is outlined in a broken line along with the corresponding group from the elementary CA rule below it. This CA rule can be equivalently summarized as an eight-bit binary string 0b10110110, a two-digit hexadecimal 0xB6, or a decimal 182. In (b), an execution of rule 182

is shown that resembles a Sierpinski triangle.

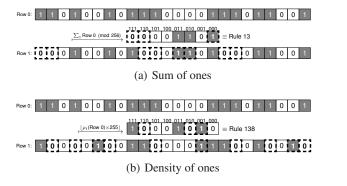
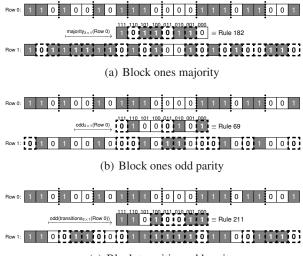


Figure 2: Example aggregation PICARD mappings. The CA rule induced by each mapping is shown between rows 0 and 1, where the triplet above each position represents the pattern of bits in row 0 that will result in the boxed bit in row 1. Positions in the rule that result in a changed bit from row 0 to row 1 have been highlighted with broken lines; the resulting toggled bits in row 1 have also been highlighted. Each mapping represents some aggregate property of the first row. In (a), the rule represents the total number of 1's in row 0, modulo 256. Because there are only 24 cells in the row, this PICARD mapping can only induce 25 different elementary CA rules. In the case of a row with more than 255 cells, every elementary rule is possible. In (b), the rule represents the density of 1's in row 0 scaled by 255.

the terms rule and macrostate interchangeably.

For example, Fig. 2 shows two arbitrary examples of microstate-to-rule mappings. Each of these two mappings extract an aggregate property of the corresponding row. In Fig. 2(a), the focal macrostate is the number of 1's in the row. In Fig. 2(b), the focal macrostate is the density of 1's in the row. For the analogous case of interacting particles, macroscopic aggregates like temperature and pressure are not typically viewed as having causal influence on the microscopic states of the system. That is, macroscopic states and microscopic states exist in two different levels of description. However, if a collection of particles is put into contact with a heat bath characterized by only its temperature, the macroscopic properties of the gas appear to drive the evolution of the microscopic system. Likewise, these PICARD aggregation mappings are simultaneously descriptive coarse grainings as well as prescriptive rules governing the dynamics of the cells. Goldenfeld and Woese (2011) argue that self-reference appears in biological systems but apparently not in physical systems because of similar reasoning - biological phenomena are emergent and may require self-reference for analysis without appealing to underlying interacting physical microstates. Self-reference therefore may be intimately related to framework suggested to characterize emergence, such as top-down causation (Davies, 2012), and may play an important role in the emergence of new organiza-



(c) Block transition odd parity

Figure 3: Example block PICARD mappings. Rules are shown just as in Fig. 2. However, the 24 bits in row 0 have been broken into 8 3-bit clusters shown with separating fences. The eight bits of the rule are then induced from a simple 3-bit predicate from each group to the corresponding rule bit. In (a), each bit in the rule is a 1 if and only if the corresponding 3-bit cluster in row 0 has a majority of 1's. In (b), the rule bit is set if there are an odd number of 1's in the cluster. In (c), the rule bit is set if there are an odd number of zero-to-one or one-to-zero transitions in the 3-bit cluster (see Table 1).

tional levels through major evolutionary transitions (Walker et al., 2012).

Aggregation mappings may have intuitive connections to statistical mechanics, but PICARD mappings may be built in entirely different ways. The block mappings in Fig. 3 are formed by clustering bits of the CA microstate into eight groups and matching each group to a predicate function that determines the bit in the rule which has the same relative position as the 3-bit group. For example, in Fig. 3(a), the 24 bits of the microstate are broken into eight mutually exclusive groups of three bits. Each 3-bit group is replaced by a single 1 if it has a majority of 1's and a 0 otherwise. In Fig. 3(b), groups are mapped into 1 if they have an odd number of 1's. Alternatively, in Fig. 3(c), the actual binary identity of the bits in each group is ignored and instead the number of transitions is used (see Table 1). That is, from left to right within each 3-bit group, a cell will transition from one level to another 0, 1, or 2 times. The corresponding rule bit will be a zero unless the 3-bit group only contains a single transition. This predicate is equivalent to the exclusive or (XOR) of the outer two bits in the microstate group. As these examples show, PICARD executions starting from the same microstate can be very different based on the chosen mapping from microstate to rule.

Microstate Block	Number of Transitions	Rule Bit
000	0	0
001	1	1
010	2	0
011	1	1
100	1	1
101	2	0
110	1	1
111	0	0

Table 1: Lookup table for transition odd parity. Reading from left to right, the number of zero-to-one and one-to-zero transitions is counted. The odd parity of this count is equivalent to the exclusive or of the outer two bits in the microstate block.

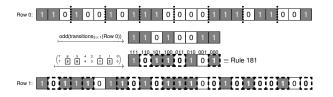


Figure 4: Block transition mapping with permutation. This PICARD mapping is equivalent to the mapping from Fig. 3(c) composed with a permutation that switches middle two bits of each nybble.

While the predicates used within block mappings may be relatively simple, the selection of microstate and macrostate bits for each block mapping adds an additional layer of complexity. For example, a single PICARD block mapping may use different predicates for each microstate block. Additionally, microstate blocks may not be mutually exclusive – they may overlap or leave some microstate bits uncovered. Even when a a single predicate is used over a set of mutually exclusive, equally sized microstate groups that cover all microstate bits, the ordering of the microstate groups need not match the ordering of the corresponding rule bits. In Fig. 4, a mapping of this sort is shown. That is, the mapping from Fig. 3(c) has been composed with a permutation. For the remainder of this paper, we will use this mapping to demonstrate the richness of an individual PICARD.

A PICARD Case Study

In the following, we consider the mapping described above in Fig. 4. Because the rule induced by each row can be viewed as a macrostate of the system, it is useful to consider both the CA microstate dynamics and the rule macrostate dynamics. In Fig. 5, executions of the PICARD mapping are shown starting from three different initial conditions. The executions of the CA are shown in the left column, and the corresponding executions through the macrostate rule space

is shown in the right column.

In general, the lower resolution macrostate executions are each visually similar to their higher resolution microstate. In Fig. 5(a), the microstate reaches a steady-state oscillation. Given that the rule that governs the microstate evolution is derived from the microstate itself, it is not surprising that the macrostate also reaches a steady-state oscillation. Similarly, in Fig. 5(b), the microstate and macrostate both reach a fixed point. However, while the CA does not reach its fixed point until generation 16, its macrostate becomes fixed in generation 14. Thus, at generation 14, this PICARD degenerates into an elementary CA under rule 44 (i.e., 0b00101100). If all generations before 14 were removed from the history of the CA, it would be indistinguishable from an elementary CA. Moreover, as shown in Fig. 5(c), this phenomenon is not restricted to only fixed points. By generation 8, the rule trajectory becomes fixed on rule 5 (i.e., 0b00000101); however, the CA trajectory reaches a steady-state oscillation where some cells are fixed and others cycle. Again, the tail of this CA execution is indistinguishable from one that would be generated under elementary CA rule 5. Consequently, this single PICARD mapping is able to represent executions from multiple elementary CA rules simultaneously while also potentially producing novel executions.

When a PICARD mapping generates an execution that is in a region of macrostate invariance, we call that execution a macroexecution. Once a macroexecution has been reached, state feedback is not required to determine the future trajectory. Thus, macroexecutions move through regions of locally elementary CA space. For the mapping explored in this section, Fig. 6 shows the relative "size" of each of these locally elementary regions. In particular, for each of the 256 elementary CA rules, all of the disconnected macroexecutions for which the rule is invariant were counted. The rules with non-zero counts were ranked and plotted as the open circles in the semilog frequency distribution in Fig. 6. The asterisks on the plot represent the number of oscillatory macroexecutions - that is, the asterisks represent the count when the fixed-point macroexecutions are removed. Thus, the highest ranking elementary CA rule includes all fixedpoint macroexecutions, but the second ranking elementary CA rule includes no fixed-point macroexecutions.

As shown by the piecewise linear broken line in Fig. 6, the highest ranking elementary CA rules have counts which peak at 2¹⁶ and are halved with each increase in rank over a significant range of rules. However, this trend is not long lasting and may simply be an artifact of the particular PICARD mapping we have chosen for this case study. In Fig. 7, the same distribution is shown on a log–log plot and shows evidence that the greater trend is Zipfian (Zipf, 1949). Words in a language tend to be Zipfian distributed in their use – the relative frequency of words is inversely proportional to their rank. Following an analogy with language, elementary CA rules act like words which are used

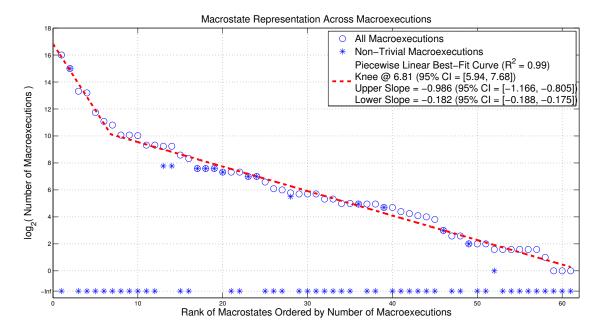


Figure 6: Macrostate representation over macroexecutions. Each macrostate may be invariant across a range of disconnected macroexecutions. Every macroexecution terminates in a fixed point or a periodic cycle of microstates that each correspond to the same macrostate. Each terminating fixed point or cycle may be reached from a number of other transient microstates which themselves correspond to that same macrostate. A complete macroexecution includes the terminating cycle and all leading transient microstates which share the same macrostate. Using the PICARD mapping from Fig. 4, the entire microstate space has been explored and every complete macroexecution has been found. Displayed here as open circles are the \log_2 -scaled counts of macroexecutions within each macrostate, omitting macrostates with no macroexecutions. The asterisks represent the \log_2 -scaled count of macroexecutions after all fixed-point macroexecutions are removed. The horizontal axis shows the rank of the corresponding macrostate when ordered by the total number of complete macroexecutions within it. A broken piecewise-linear line of best fit is shown with an upper slope near -1.

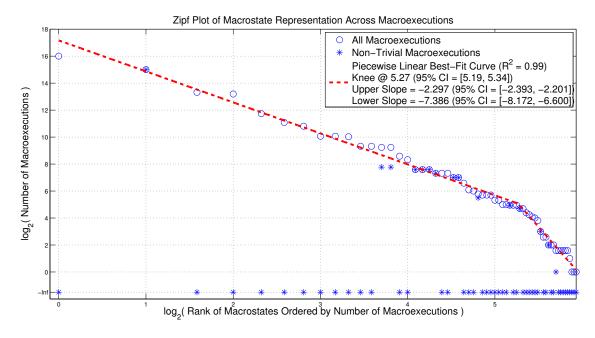
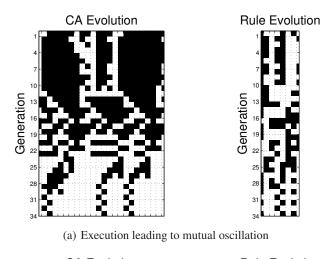
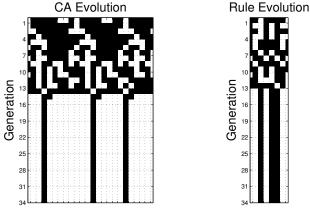
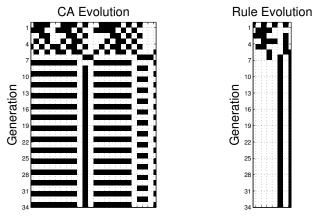


Figure 7: Zipf plot of macrostate distribution over macroexecutions. Displayed are the data from Fig. 6; however, the horizontal axis is also \log_2 scaled. The broken piecewise-linear line of best fit shows that the distribution is approximately Zipfian.





(b) Execution leading to mutual fixed points



(c) Microstate oscillations under invariant rule

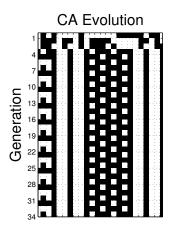
Figure 5: Example PICARD executions. Cellular automata on the left are adjacent to their corresponding rule histories on the right. Executions in (a) and (b) show CA and rule histories that have equilibria of the same type. In (c), an execution is shown where oscillations in microstate do not correspond to oscillations in rule.

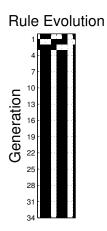
Rule	Macroexecutions	Non-trivial Macroexecutions
204 (0xCC)	65536	0
51 (0x33)	32768	32768
205 (0xCD)	10196	0
76 (0x4C)	9360	0
236 (0xEC)	3420	0
200 (0xC8)	2166	0
4 (0x04)	1792	0
12 (0x0C)	1072	0
132 (0x84)	1072	0
68 (0x44)	1040	0
140 (0x8C)	640	0
196 (0xC4)	640	0
108 (0x6C)	603	219
201 (0xC9)	603	219
232 (0xE8)	384	0
77 (0x4D)	320	0

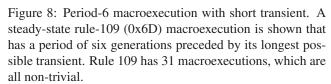
Table 2: Sixteen highest ranking macrostates.

in PICARD macroexecutions. Using a functional-dynamics framework as a model for the evolution of natural language, Kataoka and Kaneko (2000b) show how self-referencing function dynamics can act like a filter that produces a corpus of words that are each fixed points of the dynamical model. The invariant macrostate CA rules that we describe are qualitatively similar to this idea, and their representation is consistent with measured distributions of words in actual natural language.

The data from Fig. 6 have been reproduced in Table 2 for the sixteen highest-ranking macrostates. The two highest-ranking macrostates, rule 204 and rule 51, each have a number of macroexecutions that is a power of two, 2^{16} and 2¹⁵. In the case of rule 204, its macroexecutions exhaust all of the 216 microstates that map to it. Thus, rule 204 has no transients - it is a region filled entirely with fixed points. Likewise, because there are no fixed points in rule 51 and yet 2^{15} macroexecutions, each macroexecution must be a period-2 cycle. So the rule-51 region also has no transient microstates, but it is filled entirely with these cycles. Moving farther down the ranks, more microstates become available for forming macroexecutions with longer-period oscillations within them. For example, Fig. 8 shows a period-6 oscillation embedded within a rule-109 macroexecution. Because each of the 28 macrostates represents a finite number of microstates (2^{16}) , the upper bound on the number of distinct oscillations which are macrostate invariant must decrease with the average length of those oscillations. Likewise, rule 109 has only 31 macroexecutions, which are all non-trivial and thus all end on cycles of period 2 or greater. Although this mapping favors short-period





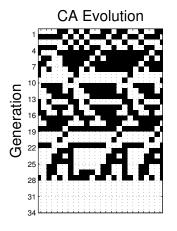


oscillations, other mappings may allow for longer cycles. For example, if the aggregate sum mapping in Fig. 2(a) is composed with a function that maps a unity sum to rule 4, then PICARD will shift each sum-1 row to the right by one position on each iteration. These shifts will not alter the sum of the row, and thus this shifting will continue indefinitely. Consequently, in this hypothetical case, the rule-4 region will contain macroexecutions with periods as long as the length of each microstate row. However, by the discussion above, rule 4 is unlikely to contain a high number of macroexecutions due to the number of microstates consumed by these long-period oscillations.

As shown above in Fig. 5(a), PICARD oscillations need not be contained within a macroexecution – microstates can oscillate across rule boundaries. Additionally, the transient components of PICARD executions can have rich structure. In Fig. 9, there is a long sequence of transient execution leading up to an eventual rule-0 fixed point. The CA appears to have significant structure visually, but the rule evolution is less predictable and shows clear path dependence. Thus, the feedback in PICARD both constrains its executions and generates novel patterns that may be fodder for further analysis.

Summary and Future Work

By feeding a conventional one-dimensional elementary cellular automaton's own state back into the iteration rule that generates its next state, we have developed a new modeling framework for investigating information control in natural systems. Although open-loop cellular automata that evolve under strictly static rules have been useful tools for studying evolution artificially, these classic frameworks are not sufficiently rich to model the dependence of fitness on cur-



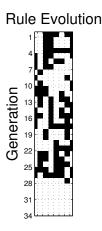


Figure 9: Fixed point with long structured transient. A rule-0 fixed point is shown that is preceded by a very long transient that appears to have significant visual structure.

rent state. Moreover, by adapting the methods of evolving cellular automata so that PICARD mappings can be the targets of artificial selection, there is a potentially richer set of behaviors that can be selected for.

We have shown how state feedback can generate self-reinforcing regions of behavior. Thus, this framework provides a model of how functional diversity can be embedded within a single automaton. This functional diversity is similar to the diverse differentiation possibilities for cells in living organisms. Moreover, as evidenced by the distribution of macrostates over macroexecutions, this framework apparently shares similarities with the processes responsible for the generation of natural language.

A possible criticism of PICARD is that it introduces nonlocal effects to cellular automata. Traditional CA are built from the assumptions that cells update based entirely on local information and information about neighbors. By feeding the state of an entire row back into the CA rule, this locality assumption is broken. A complete reductionist model of the dynamics of life would necessarily have to incorporate dynamics of both the focal entity and the environment around it – despite any significant differences in time scales. By feeding the state of the focal entity back into the rules that govern how the entity evolves, we avoid these complications and still allow for entities to alter and be altered by their environment. Thus, just as coarse-grained descriptions have utility in modeling physical phenomena, non-local effects in PICARD mappings can be viewed as useful coarsegrainings of otherwise less tractable models.

As PICARD Implements CA Rules Differently, there are myriad directions for future work – including directions which parallel investigations from conventional cellular automata and directions which are specific to PICARD self-reference. At a minimum, the distribution of macrostates over macroexecutions needs to be investigated for a wide

variety of PICARD mappings to determine whether the Zipfian distribution is widespread. Although it is attractive to view the PICARD automata in this paper as a mosaic of different elementary CA that each govern small patches of local elementarity, it is unlikely that such regions exist that contain executions with the relatively open-ended complexity of open-loop elementary CA patterns. However, PICARD mappings have the potential of generating new patterns and new functionality that may be out of reach of conventional CA's. Alternatively, PICARD mappings may have the ability to increase the robustness of elementary CA function by reducing the sensitivity to minute but functionally inconsequential changes in initial condition. Extending PICARD mappings to be able to map from prior microstates may allow for incorporating functionality normally associated with higher-dimensional cellular automata. For example, Langton's loops (Langton, 1984) are able to self replicate because growing patterns in space can turn and interact with earlier patterns. A PICARD mapping can similarly connect iterations across time and draw a connection between two-dimensional self replication and one-dimensional pattern generation. In general, an important future direction is to connect one-dimensional PICARD insights to higher dimensional cellular automata.

Acknowledgements

Kunihiko Kaneko and Larissa Albantakis provided useful feedback on earlier versions of this work. Interaction with them was possible due to a workshop on Information, Complexity, and Life organized by the BEYOND Center for Fundamental Concepts in Science at Arizona State University. Andrea Richa also provided helpful comments regarding the presentation of this material. This project/publication was made possible through support of a grant from Templeton World Charity Foundation. The opinions expressed in this publication are those of the author(s) and do not necessarily reflect the views of Templeton World Charity Foundation.

References

- Breukelaar, R. and Bäck, T. (2005). Using a genetic algorithm to evolve behavior in multi dimensional cellular automata: emergence of behavior. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation (GECCO '05)*, pages 107–114, Washington, DC, USA.
- Das, R., Mitchell, M., and Crutchfield, J. P. (1994). A genetic algorithm discovers particle computation in cellular automata. In *Proceedings of the Third Conference on Parallel Problem Solving from Nature*, volume 866 of *Lecture Notes in Computer Science*, pages 344–353. Springer.
- Davidson, E. H. (2010). The Regulatory Genome: Gene Regulatory Networks In Development And Evolution. Academic Press.
- Davies, P. C. W. (2012). The epigenome and top-down causation. *Interface Focus.*, 2(1):42–48.

- Goldenfeld, N. and Woese, C. (2011). Life is physics: evolution as a collective phenomenon far from equilibrium. *Annu. Rev. Condens. Matter Phys.*, 2(1):375–399.
- Hofstadter, D. R. (1979). Gödel, Escher, Bach: an Eternal Golden Braid. Basic Books, Inc.
- Hordijk, W. (2013). The EvCA project: a brief history. *Complexity*, 18(5):15–19.
- Kataoka, N. and Kaneko, K. (2000a). Functional dynamics. I: articulation processes. *Phys. D: Nonlinear Phenom.*, 138(3–4):225–250.
- Kataoka, N. and Kaneko, K. (2000b). Natural language from function dynamics. *Biosystems*, 57(1):1–11.
- Langton, C. G. (1984). Self-reproduction in cellular automata. *Phys. D: Nonlinear Phenom.*, 10(1–2):135–144.
- Mitchell, M., Crutchfield, J. P., and Hraber, P. T. (1994). Evolving cellular automata to perform computations: mechanisms and impediments. *Phys. D: Nonlinear Phenom.*, 75(1–3):361–391.
- Schlitt, T. and Brazma, A. (2007). Current approaches to gene regulatory network modelling. *BMC Bioinformatics*, 8(6).
- Walker, S. I., Cisneros, L., and Davies, P. C. W. (2012). Evolutionary transitions and top-down causation. In *Proceedings of the Thirteenth International Conference on the Simulation and Synthesis of Living Systems (Artificial Life 13)*, volume 13, pages 283–290, East Lansing, Michigan.
- Walker, S. I. and Davies, P. C. W. (2013). The algorithmic origins of life. *J. R. Soc. Interface*, 10(79).
- Wolfram, S. (2002). A New Kind of Science. Wolfram Media.
- Zipf, G. K. (1949). *Human Behavior and the Principle of Least Effort*. Addison-Wesley Press, Oxford, England.

There can be Only One: Reversible Cellular Automata and the Conservation of Genki

Nathaniel Virgo¹ and Takashi Ikegami

Ikegami Laboratory, University of Tokyo ¹nathanielvirgo@gmail.com

Abstract

Reversible cellular automata (RCAs) are a special class of cellular automata with some very distinctive properties. We present a novel observation regarding a certain class of RCAs, a class that includes Norman Margolus' "Critters" rule. From a broad range of initial conditions, this class of cellular automata converge to a state in which all the structures in the system are periodic (the equivalent of blockers and blinkers in Conway's life), with the exception of a *single* glider.

The glider that remains is immortal. It follows from a generic property of RCAs that the last glider in the system cannot be destroyed, and its motion cannot enter a periodic cycle. On colliding with a periodic structure, the last glider may change direction or turn into a different type of glider. Very occasionally it will transform into two gliders for a period of time, but when these collide the result is likely to contain only a single glider again. We give some intuitive explanations for why the system converges to a state with only one glider, rather than many.

It seems relatively easy to construct systems with this single-glider property using block CA rules. We give an example where cells can take a number of different colours, and gliders must contain at least three colours (including the background one). When a glider collides with a periodic structure, the new glider resulting from this collision may be composed of different coloured cells than the original one. Thus, some essential organisational property has been transferred from one set of coloured tiles to another. We call this property "genki", after a Japanese word meaning health or vitality. We speculate on how it might be formally defined and whether it is applicable to RCAs in general.

Some cellular automata have the interesting property of being reversible, meaning that the previous state can always be determined from the current state. Such systems can never "destroy information", which constrains their dynamics in interesting ways. A reasonably well-known example is Norman Margolus' "Critters" (Margolus, 1999), the rules for which are shown in Figure 1. Here we present a slight variation on the Critters rules (also in Figure 1), which we call "Highlander", after a movie in which immortals battle to the death until only one remains (Mulcahy, 1986).

In addition to being reversible, Critters and Highlander share the property that they conserve the number of white

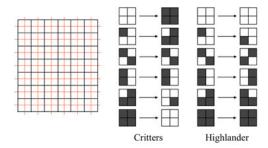


Figure 1: (Left) Critters and Highlander are block cellular automata. On even time steps the grid is divided into blocks as indicated by the black lines, and on odd time steps as indicated by the red lines. (Right) on each time step, the rules are applied to each block independently. Critters restores the original number of white cells after two time steps, whereas Highlander preserves the number of white cells directly. Both sets of rules are reversible because, for each block, the previous state can be uniquely determined given the current state.

and black cells. They have similar dynamics, but with a key difference: gliders emerge less readily from random initial configurations in Highlander than they do in Critters.

As pointed out by Margolus, reversible CAs have the property that if two gliders (or a glider and a blinker) collide, the resulting pattern cannot remain spatially bounded. If it did, it would have to go into a periodic orbit, violating reversibility. Thus, at least one glider must always emerge from a collision. We call the property of being able to produce one glider "genki". Gliders transfer genki from one region to another, and genki is always preserved in collisions, although the number of gliders is not.

Because gliders are difficult to produce in Highlander, there is a tendency for only one glider to be created from the debris of a two-glider collision. An example of this is shown in Figure 2. This has a curious consequence: Starting from an initial condition with only a small proportion of white pixels, which are initially concentrated in one place, the system converges to a state in which only a single glider exists.

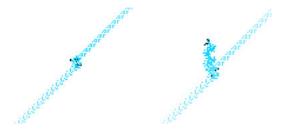


Figure 2: A head-on collision between two gliders. White pixels that were black a multiple of 60 time steps ago are shown in shades of blue. (Left) after the collision, a somewhat chaotic state is produced. However, since this state cannot be periodic, it must eventually emit at least one glider. (Right) 3360 time steps later, a glider is visible moving upwards from the point of the collision, with some non-moving debris left behind.

This glider bounces off stationary pixels in a pseudo-random way, gradually dispersing them throughout the space. An example of these dynamics (on a 400×400 grid with periodic boundary conditions and an initial square of 11×11 black pixels) is shown in Figure 3.

We can make sense of this from the point of view of statistical mechanics. The reversibility property means that a finite system must eventually return to its initial state, but this may take a huge number of iterations. The initial state we choose is a highly atypical one, but the system must spend the vast majority of its time in much more typical states. Our initial conditions have a proportion of black cells of about 0.75%. A typical configuration with this proportion of black cells will not contain a single glider, because gliders require several black cells to be near each other. (We have not observed any gliders with fewer than 4 black cells.) However, the conservation of genki prevents these states from being reached: the final glider cannot be destroyed. Thus, we hypothesise, the system will spend the vast majority of its time in the most typical type of state that it can access: one where the majority of white cells are scattered uniformly, with the initial genki contained in a single glider.

It seems relatively easy to create RCAs with the Highlander property. We suspect that Critters itself has it, but it takes longer to reach the single-glider state because new gliders are formed more readily in Critters. In particular, we have produced a multi-coloured version of the Highlander rules, where cells can have n different states, and the rule applied depends only on the number of distinct states in a given block: if there are two distinct states in a block (including black), it is rotated 90° ; if there are three then it is rotated 180° , and if there are four then it is rotated 270° . (Many variations of these rules work equally well.) With these rules a glider may change its constituent colours in a collision, as well as its speed and direction. (Figures illustrating this are omitted for reasons of space.)

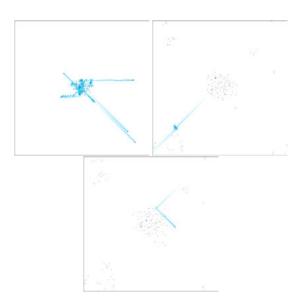


Figure 3: (Top left) t=14400. Note that there are several types of glider, which move at different speeds in different directions. (Top right) t=1374240; the last two gliders have collided. A single glider will result from the collision. (Bottom) at t=6100560 there is still only a single glider in the system. Collisions may transform it into a different glider type and/or change its direction but are very unlikely to produce a second glider. These collisions have moved the black pixels slightly closer to a uniform distribution.

This abstract is not intended as anything other than an observation of an interesting phenomenon that we believe has not been noted before. However, careful study of reversible cellular automata might help us to understand complex phenomena in physics and chemistry. The laws of quantum mechanics obey unitarity, which is a close cousin of reversibility, and we therefore expect properties of reversible cellular automata to be shared by physical phenomena on the microscopic level. It is not clear what physical concept, if any, corresponds to our notion of genki. However, we note a resemblance between gliders in RCAs and particles in quantum field theory. In both cases, collisions may result in a variety of different outcomes, which are constrained by conservation laws. On a slightly larger scale, the gliders in these systems resemble radicals in chemistry, in that a reaction which destroys a radical also tends to create a new one. Continued study of these artificial systems may provide new insights into the physics that underlies living systems.

References

Margolus, N. (1999). Crystalline computation. In Hey, A., editor, *Feynman and Computation*, chapter 18. Perseus Books.

Mulcachy, R. (Director). (1986). Highlander. Cannon Films.

A Hybrid Off/On-Lattice Model of Emergence and Maintenance Autopoiesis

Olivier Wang^{1,2}*, René Doursat^{1,3} and Paul Bourgine^{1,3}

¹Ecole Polytechnique, Paris, France – European Erasmus Mundus Master's in Complex Systems Science (MCSS)

²IBM France Lab, Paris, France

³Complex Systems Institute, Paris Ile-de-France (ISC-PIF), CNRS UPS3611, Paris, France

*corresponding author: olivier.wang@polytechnique.edu

Abstract

We propose an original 2D agent-based model of biological "autopoiesis", the process by which a cell creates and continuously regenerates itself, considered one of the defining characteristics of life. In the space of our simulations, the positions of free molecules are continuous, and polymerized membrane components are regularly arranged (plus noise). While autopoiesis commonly refers to the self-driven maintenance of a system, we also follow Varela's historical study of emergence, and show that the same model can account for both selfperpetuation and self-formation—a step toward uniting the three main perspectives on life: origins, autopoiesis, and replication. Exploring different initial and environmental conditions, we observe that destructive reactions are important for the survival of our autopoietic system, and evaluate their impact on its lifespan. The tendency of cells to form spurious outgrowths is counteracted by moderate decay of the membrane.

Introduction

The study of life and consciousness construed as high-level, abstract systemic properties, beyond their biochemical composition, has led to the concept of "autopoiesis", a term coined by Maturana and Varela (1973) to refer to the self-creation and self-repair abilities of organisms. The initial idea of defining living systems as fundamentally autopoietic is well accepted today, and has also been broadened to include cognitive systems (Bourgine and Stewart, 2004). Yet, autopoiesis is still at the center of a long-lasting debate (Fleischaker, 1992), where critics object to the overly theoretical nature of the concept and deem self-referentiality without external references meaningless (Swenson, 1992). By contrast, other fields such as sociology (Luhman, 1986) have embraced autopoietic thinking and imported it into their research. The original 1973 definition stated that:

An autopoietic system is a machine organized (defined as a unity) as a network of processes of production (transformation and destruction) of components that produces the components which: (i) through their interactions and transformations continuously regenerate and realize the network of processes (relations) that produce them; and (ii) constitute it (the machine) as a concrete unity in the space in which they exist by specifying the topological domain of its realization as such a network.

Many variants have been proposed since. We base our model on Bourgine and Stewart's more recent approach (2004):

An autopoietic system is a network of processes that produces the components that reproduce the network, and that also regulates (from inside) the membrane conditions necessary for its ongoing existence as a network.

We will also include the less explicited notion that, since the components are what reproduces the network, the processes cannot be based on global variables or global properties of the system, but must rely on purely local interactions.

These different viewpoints on autopoiesis have produced various computer models. Agent-based simulations, where a large number of discrete units are updated sequentially or synchronously, have been the tool of choice to illustrate the concept of autopoiesis since its inception. The first attempt (Varela et al., 1974) used cellular automata (CA), and was pursued and extended by Zeleny (1977) who formalized the computational model and explored other scenarios, in particular ones involving more catalysts in the environment. A later analysis of this original model by McMullin and Varela (1997) pointed out that, although it was missing a critical 'additional interaction", the algorithm did achieve autopoiesis when membrane creation was inhibited in its interior neighborhood. Further developments have also allowed movement of the simulated membrane (Breyer et al., 1998; McMullin and Groß, 2001).

The majority of other simulated autopoietic systems have been confined to a CA grid, such as Beer's analysis of a glider in the Game of Life (Beer, 2004), or the 3D lattice artificial chemistry of Ono and Ikegami (1999, 2000, 2001) based on hexagonal units, as was Sirmai's 2D "morphautomaton" (2011, 2013). Beer proved that there can be other types of autopoietic systems than molecular ones imitating the biochemistry of life. He also emphasized that proper definitions of the system and its boundary are essential when determining whether a system is autopoietic or not. Ono and Ikegami showed that particle interaction models can exhibit autopoietic behavior, and hypothesized about the structure of early life forms on Earth. Sirmai illustrated how the self-maintenance mechanisms of an autopoietic system could also give rise to self-replication when adding a few interactions.

The relative lack of autopoietic simulations in continuous space can be partially explained by the computational cost compared to discrete models. Whereas, more generally, the field of artificial chemistry has produced continuous implementations (Hutton, 2007; Ono and Ikegami, 2001: Chap. 3) these have not addressed autopoiesis per se. Moreover, the

continuous case is only described qualitatively, while quantitative results are generally produced by a discrete approximation on a lattice. A review of computational models of autopoiesis was written by McMullin on the 30th anniversary of the concept's discovery (McMullin, 2004).

In this paper, the molecular dynamics is an artificial chemistry that happens for the most part in continuous 2D space. A discrete lattice component is still present to constrain the locations and behavior of membrane particles, once they have bound to each other. We take inspiration from Bourgine and Stewart (2004) to define the reactions, and simplify the system so that it involves fewer classes of components. Two possible outcomes are analyzed, one corresponding to the usual sense of autopoiesis as *self-maintenance*, and the other representing the ability of the system to create itself from a single element, or *emergence*, which is closer to the etymology of the term. The latter is necessary to the former and to self-reproduction.

A Model of Cell Autopoiesis

Particles, Membrane, and Environment

We consider a continuous 2D environment, the world. It is filled with particles animated by Brownian motion (i.e. which follow straight trajectories between two collisions), called substrate particles and denoted by S (pink disks in Figs. 1-4). Their density, i.e. average number per lattice square, is set to 0.4. In the world, we observe the creation and/or survival of an autopoietic system, a cell, made of two other types of particles, monomers M (yellow or green disks) and component particles C (blue disks). The cell is characterized by a semipermeable membrane arising from monomers binding together. Monomers have a diameter of 1 and are locked into lattice positions regularly spaced by the same unit, with a small Gaussian vibration of width ε . The membrane serves as a boundary for the system and can self-repair as a whole. Monomers randomly decay into *waste* particles W (orange disks), at a low rate δ , thus gradually damaging the membrane over time. Counteracting this, the membrane is also repaired by component particles that are present in the enclosed space and transform into monomers. Details are explained below.

The monomers of the membrane are permeable to the small substrate particles; they do not impede their movement. On the other hand, the substrate, component and waste particles all collide and interact with one another in a way similar to gas molecules. Monomers have an orientation vector (Fig. 2a) to mark the inside and the outside of the membrane, i.e. the two half-spaces created by a dividing line orthogonal to that orientation and running through the center of the molecule. We also say that the inside is "behind" the monomer, and the outside is in the "front". The local curvature of the membrane is a fixed property of the monomers, arising from their internal molecular structure. It is modeled here by a constant angle α between two neighboring orientation vectors.

Upon colliding, these various particles react in different ways depending on their type. We consider in this model three key interactions resulting in the creation or destruction of particles (Fig. 2): the *synthesis* of components, the *repair* of membrane monomers, and the *decay* of monomers.

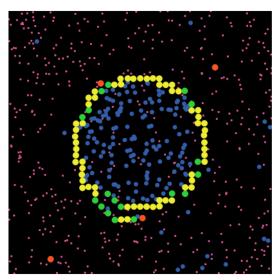
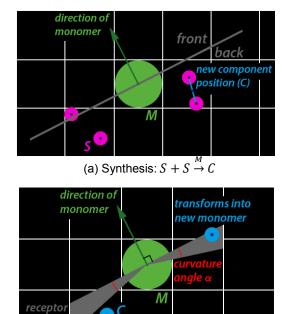


Figure 1: Example of autopoietic cell. *Pink*: substrate particles *S*; *blue*: components *C*; *yellow*: doubly bonded monomers, M_2 ; *green*: simply bonded, M_1 , or unbonded monomers, M_0 ; *orange*: waste particles, W. In the top-left quadrant of the membrane, two pairs of neighboring M_1 's exhibit holes of size $\sqrt{2} - (2 \times 0.5) = 0.41 \pm 2\varepsilon$. Near the upper hole, one M_2 has just decayed into a W. In the top-right quadrant, there is a larger hole of size $\sqrt{5} - 1 = 1.24 \pm 2\varepsilon$. At the bottom-left, the system has formed two layers, starting a spiral shape. Thus it is not autopoietic because its membrane is not closed. Time tick: $t_i = 350$. Parameters: membrane curvature angle $\alpha = 8^\circ$, M-decay rate $\delta = 0.05\%$, C-saturation level c = 2 and radius r = 10, M-vibration width $\varepsilon = 0.05$.

Synthesis: $S + S \xrightarrow{M} C$ Two substrate particles can produce one component particle under specific conditions: the reaction must be catalyzed by a nearby monomer M and only if it happens "behind" it (Fig. 2a). Moreover, it should not be inhibited by too many other C's in the local environment: the density of C particles in a given radius r should stay below a saturation level c. If these conditions are met, then the two S particles disappear and a C is put in their place, with averaged speed and direction. These two conditions are local versions of Bourgine and Stewart's model (2004).

Repair: $C + M_1 \rightarrow M + M_2$ This reaction transforms a freely moving component into a monomer that integrates the chain of monomers forming the membrane. This can involve either filling a hole that has appeared in an existing membrane ("maintenance autopoiesis"), or building a new membrane ("emergence autopoiesis"). Thus, by "repair", we also mean the series of reactions that make a single M-seed grow into a complete autopoietic system. Like "synthesis", it also requires certain local conditions to be fulfilled: C particles transform into M only when they are close to a membrane hole or extremity, i.e. a "simply bonded" M, a monomer with only one neighbor, denoted by M_1 (in green). In addition, C must also be located within a specific arc of space (a 2D cone) in relation to the neighboring M, which we call receptor arc (Fig. 2b). It is defined as the domain "behind" M where the angle between the space-dividing line and the MC segment is smaller than the curvature angle α .



(b) Repair: $C + M_1 \rightarrow M + M_2$

does not transform

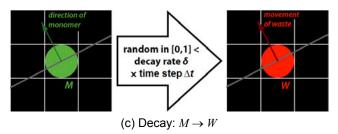


Figure 2: The three fundamental reactions of the model. (a) "Synthesis": two S particles (pink) located in the same square can create a C (blue) if an M (green) is present in a neighboring square and if they are "behind" it (verified on the right, not on the left). (b) "Repair": only a C located in a square adjacent to an M and inside the receptor arc (gray area) may transform into a new M, added to the membrane. (c) "Decay": with low probability at each iteration, any M can become a waste particle W, which receives an initial speed in the direction of M's vector (and norm 10 here).

Decay: $M \rightarrow W$ This is the simplest reaction, as it involves a single monomer randomly becoming waste (Fig. 2c). The probability of decay per time unit, δ , is constant and uniform over all M's; it is independent from the age of the M particle.

Physics simulation

Our simulated environment does not use a physics engine, unlike other artificial chemistry works, but was more simply implemented in the NetLogo platform (Wilensky, 1999). Although the disk-shaped particles have floating number coordinates, their collisions are not modeled by solid bodies. Instead, we repurposed a "dynamic billiard" gas model that partitions the world into square domains where particles interact. To ensure the detection of collisions, the duration of one time step was dynamically adjusted so that no particle could

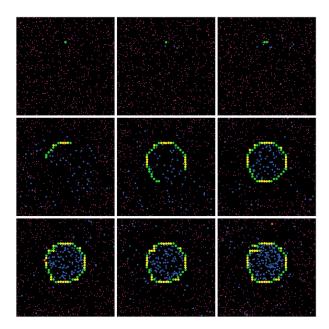


Figure 3: Example of "emergence autopoiesis" followed by temporary "maintenance autopoiesis". *Top*: At first, a few components C are created by S particles colliding in the vicinity of the seed monomer M. Then, the C's start building a circular membrane by transforming into new M's (ticks t = 0, 0.27, 1). Emerging radius is 8 (from an angle $\alpha = 8^{\circ}$). *Middle*: The chain grows and closes itself (ticks t = 64, 126, 151). *Bottom*: For a while, the cell is autopoietic, as holes appearing from the decay of M's are filled with new ones (ticks t = 210.5, 240). Eventually, however, the membrane starts spiraling and the cell dies (tick t = 364.4). Parameters identical to Fig. 1. (A video can be seen at http://tinyurl.com/locy7qy.)

travel farther than one square at every iteration. Adding these variable time steps yields time *ticks*, denoted by t_i , where i is the iteration index. The i-th time step is $\Delta t_i = t_{i+1} - t_i$, and the total duration of a run is t_N , where N is the last iteration (typically, $t_N = 3000$ in this study, which corresponds to an average $N \approx 10000$). A similar model is available online (Pelaez, 2009), although it does not demonstrate an actual autopoietic system since it constrains the membrane to a preassigned location and allows its formation only there.

In this environment, we also limit the degrees of freedom of the membrane. Bonded M particles cannot move away from the integer coordinates they are pegged to, only vibrate around them. At every time step, a small random vector is added to the center location (see Rule 9 below). The membrane as a whole cannot deform or drift. Note that an M is only labeled "unbonded" (M_0) , "simply" (M_1) , or "doubly bonded" (M_2) by proximity to other M's, but physically it behaves the same.

Algorithm

In sum, each iteration *i* with a time step Δt_i essentially consists of the following rules and actions:

1. "Displacement" rule: recalculate the velocity vector \vec{v} of the C and W particles that bounce off an M. Update the positions of all moving particles by adding $\vec{v}\Delta t_i$. Particles that exit the world disappear permanently.

- 2. "Replenish" rule: create new S particles at the edge of the world to simulate a solution of constant S-concentration.
- "Repair" reaction rule: test pairs of neighboring C's and replace them with new M's where appropriate (see above).
- "Synthesis" reaction rule: change the properties of colliding S particles: either create one C if proper conditions are met (see above), or modify their \vec{v} if they only rebound.
- 5. "Decay" reaction rule: for each M, if a uniformly random number in [0,1] is less than $\delta \Delta t_i$, replace it with a W.
- *Increment the iteration counter:* $i \rightarrow i + 1$.
- 7. "Time update" rule: determine the length of the next time step Δt_{i+1} by polling the velocities \vec{v} of all the moving particles and making sure $\| \vec{v} \Delta t_{i+1} \|$ does not exceed 1.
- "Vibration" rule: every full unit of time, add a small random vector, with uniform angle in $[0, 2\pi]$ and Gaussian norm of width ε , to each M around its lattice position.

This algorithm is used for both types of initial condition (existing membrane and single monomer) and runs until tick $t_N = 3000$ for all the results presented in this article.

System definition, exploration, and measure

Through this model, we want to study the autopoietic properties of a class of systems defined by a closed, roughly circular membrane made of bonded particles of one type (here, monomers M) and enclosing freely moving particles of another type (here, components C). The topology of a system can be assessed by the distribution of holes in the chain of monomers composing the membrane, including their number, locations and sizes. The size h of a hole is defined as the shortest distance between two neighboring M_1 's (simply bonded monomers; green disks in figures), i.e. the diagonal distance between their centers minus 1 (twice their radius). Given a maximum hole size λ (in general 1.5, just above $\|(2, 1)\| - 1$), we calculated the number of holes n_{λ} bigger than λ and said that

- the membrane was "closed" if $n_{\lambda} = 0$,
- the membrane was "viable" if $n_{\lambda} = 1$, the membrane was "broken" if $n_{\lambda} \ge 2$.

In the last case, the presence of two holes or more created at least two disconnected pieces in the 2D world. (Naturally, these definitions would need to be modified in 3D, replacing punctual holes with lines or closed paths on the surface of the membrane.) Note that, since we only considered holes larger than 1.5, a closed or viable membrane could still contain several smaller holes (typically of size 0.4 or 1, rarely 1.24), which were called "pores". This is a fundamental characteristic of real-world cells and other autopoietic systems: while presenting a well-formed and stable membrane structure, they also kept a partial "openness" to the environment, letting material flow inward and outward through small channels.

In addition, we verified that the membrane remained roughly circular, i.e. did not form a spiral as in Figs. 3-4 or an open thread. These unwanted structures did not necessarily contain regular holes—or they could be said to contain M_1 's on the edge of "infinite" holes. For this, we checked that the virtual spokes connecting each M to the center of the world were not passing through any other M particle.

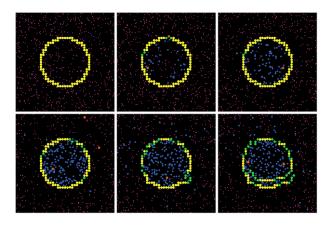


Figure 4: Example of temporary "maintenance autopoiesis". The system is initialized with a circular membrane of radius 10 (composed of 80 M's). Small holes are repaired in the beginning (top: ticks t = 0, 1, 6), more holes appear later (bottom: tick t = 46) and the membrane eventually breaks down and spirals (ticks t = 201, 369). Parameters same as Fig. 1, except for a higher decay rate $\delta = 0.15\%$. (Video at http://tinyurl.com/qcofya4.)

Among the various parameters of the model, we chose to explore two in particular: the M-decay rate δ (at which M's transform into W's) and the C-saturation level c (beyond which the synthesis of C's is inhibited). The other main parameters were fixed, with following values: curvature angle $\alpha = 8^{\circ}$, C-saturation radius r = 10, M-vibration width $\varepsilon = 0.05$. The outcome of simulation was evaluated differently in the two scenarios studied here: for maintenance autopoiesis, we measured the "lifespan" τ of viable cells and for emergence autopoiesis, we counted the closed cells together with the number and size of holes in their membrane. A summary of the setup follows. Detailed results are presented in the next section

Maintenance autopoiesis, taken separately from emergence autopoiesis, refers to the survival of an existing system starting from a fully closed membrane, with no hole of any size (Fig. 4). Our initialization used a discretized circle or radius 10 (composed of 80 M's) located in the center of a world that was filled with S particles only. The results varied from very short to very long lifespans. We measured these durations using the topological criterion based on a continally updated list of holes. Given the maximum hole size λ , the lifespan τ was set to the first tick when either the membrane was broken (the number of larger holes reached 2) or the cell started forming a spiral (the hub-and-spokes test was positive).

Emergence autopoiesis, on the other hand, is what we expect to happen when starting from a single M particle. Under the right conditions, an M catalyzes the reaction of S particles on its "back", producing C's that later transform into new M's and so on. This creates a growing chain that should eventually close and form an autopoietic system. As in the maintenance case, the decision as to whether the simulation produced a stable system depended on an assessment of the membrane topology. In the emergence case, we used the same threshold value $\lambda = 1.5$ but computed the number of closed cells only. We also screened out spiraling membranes.

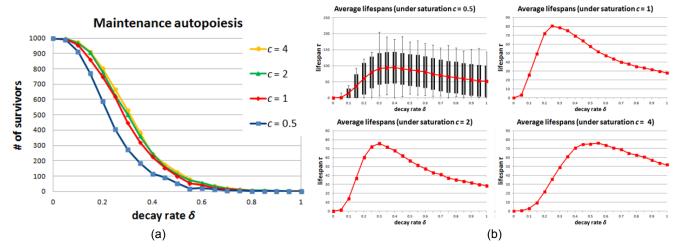


Figure 5: Statistical analysis of maintenance autopoiesis. (a) Number of surviving cells at t_N = 3000 as a function of M-decay rate δ under four different C-saturation levels c. A cell is a "survivor" if its membrane contains at most one hole larger than 1.5. Starting from 1000 cells, the half-population survival level 500 was reached for δ between 0.22% and 0.32%. (b) Average lifespans τ of the non-survivor cells as a function of δ under the same four values of c. Top-left: boxplot showing the standard deviation and extremes of each distribution (see Fig. 6). Compared to survivors, for which $\tau \ge 3000$, non-survivors were wiped out relatively quickly, from immediately ($\tau = 0$) to about $\tau = 200$. More importantly, the lifespan reached a peak for optimal values of δ , between 0.25% and 0.55% depending on c, implying that some decay helped the cells in their regeneration process. Other parameters were the same as Fig. 1.

Results

Maintenance autopoiesis

A statistical analysis of the model was performed on 1000 runs for each set of parameters. Each successful run lasted until tick t_N = 3000. The M-decay rate δ was varied from 0% to 1% by increments of 0.05%, and four different values of C-saturation level were tried: c = 0.5, 1, 2, and 4 (Fig. 5). First, they confirmed that our model supported the "maintenance" form of autopoiesis, as a significant number of runs ended with closed or viable cells—which are referred to as "survivors" in this case. Although no upper limit was put on the size of the only allowed hole larger than λ (for the cell to be at least viable), empirically we never observed holes larger than h = 4.5, and such extremes occurred rarely. Thus there was no massive "unknitting" of the membrane.

The most striking observation is that the number of survivors started falling sharply for δ values as small as 0.1% and almost vanished at 0.7% (Fig. 5a). The half mark of 500 cells, roughly corresponding to the inflexion point of the curve, was reached by an average δ comprised between 0.22% (for c=0.5) and 0.32% (for c=4). This is intuitively reasonable, as the system cannot be autopoietic with too much decay or too little new material, and an increase in the former should be compensated by an increase in the latter to maintain the cell membrane.

A better insight into the behavior of the model could be gained by analyzing the average lifespan τ of non-survivor cells (Figs. 5b-6). Compared to survivors, which by definition reached at least τ = 3000, non-survivors died early in the simulation, generally before τ = 200. It means that autopoietic cells that have lasted beyond a certain time limit can expect to maintain themselves virtually forever. More interestingly, we found that, among the non-survivors, τ did not monotonously

decrease with δ , as it could be expected on first thought (i.e. the more decay, the earlier the membrane should collapse). Instead, τ first increased to a maximum comprised between 75 and 95 units of time (depending on c) for the lower range of δ values, then decreased again but more slowly. The maximum point was reached for an optimal δ between 0.25% and 0.55%, a range similar to the inflexion point of the survivors' curve.

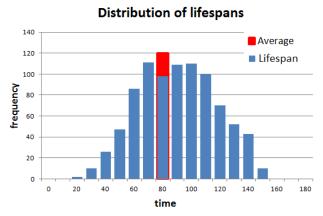


Figure 6: Distribution of lifespans τ . This histogram was obtained from 1000 runs under a C-saturation level c=4 and an M-decay rate $\delta=0.5$. It would correspond to a boxplot placed in the middle point of the bottom-right curve of Fig. 5b (average τ of 75; bins are labeled here by their upper bound). As this batch produced 126 survivor cells, only 874 of 1000 were used to compute this histogram. Other parameters same as Fig. 1.

This seemed to indicate that *some disruption of the membrane was necessary* for the cells to express their autopoietic abilities. One explanation could be that when the membrane is initially disrupted through the decay of a few *M* particles

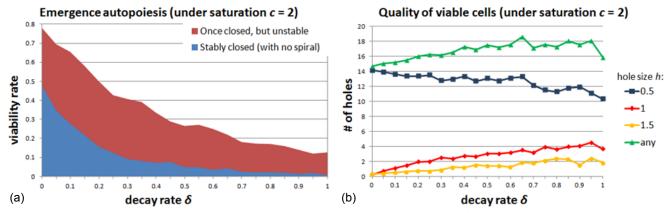


Figure 7: Statistical analysis of emergence autopoiesis. (a) Total number of "closed" newborn cells (as a fraction of the initial population of 1000). *Blue area*: cells that are still closed at the end of the simulation ($t_N = 3000$); $red\ area$: cells that reached a closed state but were unstable and did not survive. (b) Distribution of holes in the membrane of stable cells, plotted according to their size h. Three categories were defined: $0.5 \le h < 1$ ("category-0.5" holes), $1 \le h < 1.5$ ("category-1" holes), and $1.5 \le h$ ("category-1.5" holes). *Green curve*: total number of holes (sum of the other three curves).

becoming W's, which drift away and create holes, the repair process often has a tendency to generate a pathological spiral configuration—but, precisely, this spiral is less likely to grow if it is itself disrupted by a moderate level of M-decay. In short, δ would need to be sufficiently high to destabilize spurious (and ultimately lethal) repair structures, but not as high as to riddle the membrane with too many holes that cannot be repaired quickly enough. This is a form of "creative destruction" that is reminiscent of cancerous cells.

Emergence autopoiesis

Next, we investigated the self-creation of well-formed autopoietic systems from a single monomer M, based on the same dynamics and parameters as maintenance autopoiesis. As before, results were collected from 1000 runs over a total duration of 3000 time units, and by varying δ from 0% to 1%. The saturation level c was set to a constant 2 (Fig. 7). They showed that our model also supported the "emergence" form of autopoiesis. Here, we only counted the number of cells capable of building a closed membrane, i.e. with no hole larger than λ . We distinguished between cells that were still stably closed at $t_N = 3000$ and cells that were once closed but unstable and did not survive until the end of the simulation (Fig. 7a). Overall, the number of stable newborn cells was relatively low, starting at less than 500 (half the initial population) and dropping under 50 past δ = 0.5%. The total number of newborn cells, stable and unstable, was more significant, going from 800 to 250 within the same range.

In this emergence scenario, however, there was no "creative destruction" tendency: starting from 0%, a moderate increase in δ did not produce more stable cells. This is possibly due to the fact that, unlike self-perpetuation, self-creation does not involve the repeated outgrowth of small spiral "barbs" on top of a circular membrane (defects which then need to be erased by noise), but produces a more consistent development of a longer structure—ending up either in a circle or a spiral. Moreover, the range of δ values favorable to the self-creation of a closed membrane from scratch might not

overlap with the range of δ values that help an existing membrane regenerate itself. Therefore, when starting from a single M, and once δ has been set, it seems that the system can only bifurcate between a closed circle state and an open spiral state, but not transition from the latter to the former. A more complete chart of the system's "phase space" would be needed to confirm these hypotheses.

We also assessed the topology of stable cells by analyzing the distribution of holes in their membrane. Fig. 7b plots the number of holes whose size falls into three categories delimited by h = 0.5, 1, and 1.5, together with the total number of holes (of any size). While the number of category-1 and category-1.5 holes increased with the decay rate δ , the number of category-0.5 holes decreased. This could be an artifact of the simulation caused by the "receptor arc" of angle α limiting the conversion of C's into M's and making it more difficult for smaller holes to be repaired. Higher decay rates limit the occurrence of such cases but, naturally, add larger holes, too.

Discussion

We have proposed a minimal model of cell autopoiesis in continuous 2D space inspired by artificial chemistry. It contains three molecular types, a substrate, "components", and waste particles, which freely move in the environment, collide and react under specific conditions. Molecules of a fourth type, the monomers, bind to each other and occupy crystallike lattice positions, around which they vibrate, forming the cell's membrane. Results show that, under certain sets of parameters, our model is able to reproduce not only the traditional "maintenance" form of autopoiesis, but also build a new cell from scratch, displaying "emergence" autopoiesis.

Whereas other autopoietic models have been based on discrete cellular automata of various types (Varela et al., 1974; Zeleny, 1977; Breyer et al., 1998; McMullin and Groß, 2001; Sirmai, 2013) or autocatalytic sets of binary strings (Banzhaf, 1994), we have implemented a somewhat more realistic "dynamic billiard" agent-based simulation, similar to the molecular models used in microcanonical statistical me-

chanics or in artificial chemistry (Ono and Ikegami, 2001: Chap. 3), and from which we could collect meaningful quantitative results. Most of the simplifications we adopted were intended to reduce the computational cost, in particular relying on a grid approximation to detect collisions faster. One especially contrived part of the model, however, is the absence of movement of the bound monomers (beside vibration), and the consequent rigidity of the membrane, which may not deform or drift once pegged to the underlying lattice partition of space. As for the curvature angle and receptor arc presiding over membrane polymerization (Fig. 2b), they can be interpreted as an abstract reflection of the internal geometric or "conformational" properties of monomer molecules.

The fact that a cell in our model can both maintain and create its own organization from a chaotic environment is a contribution toward uniting the three main perspectives and modeling approaches to life: *origins* (Rasmussen et al., 2004), *autopoiesis*, and *self-replication* (Langton, 1984; Sayama, 2000). The data obtained from our model further suggests that there is an optimal decay rate for autopoietic systems. Too much destruction of the membrane breaks the system down, while too little allows inevitable imperfections triggered by the environment's molecular randomness to thrive and take over the cell. This observation could be relevant to the study of oncological perturbations, or the early chemical environments favorable to the emergence of biotic forms.

In conclusion, the experiments presented here could provide another foundation on which to build more elaborate and realistic autopoietic models in continuous space. While we have already started working on an extension to a 3D world, other improvements should also focus on refinining the molecular mechanisms (collisions, geometry-based interactions and reactions; Fernández et al. 2012), and inserting some genetic information, implicitly or explicitly, to let the system evolve toward self-replication. The "emergent autopoiesis" side of our model also provides another illustration of the morphogenetic engineering viewpoint on self-organized complex systems (Doursat et al., 2012, 2013), for which the onset of order can be much more than random, "texture"-like patterns (stripes, spots, waves) and exhibit strong architectural features, too (nonhomogeneity, reproducibility, programmability). These morphogenetic capabilities have spontaneously evolved in nature over millions of years and could now be artificially accelerated and put to practical applications by human inventiveness. Swarm chemistry (Sayama, 2009), the growth of nontrivial shapes from mixed particle species, and synthetic biology, the design and control of "bioware" systems, are such examples at the molecular and cellular scale.

References

- Banzhaf, W. (1994). Self-organization in a system of binary strings. Proceedings of Artificial Life IV, pp. 109-118.
- Beer, R.D. (2004). Autopoiesis and cognition in the game of life, *Artificial Life*, 10:309-326.
- Bourgine, P. and Stewart, J. (2004). Autopoiesis and Cognition, *Artificial Life*, 10(3):327-345.
- Breyer J., Ackermann J. and McCaskill J. (1998). Evolving reactiondiffusion ecosystems with self-assembling structures in thin films, Artificial Life, 4:25:40.
- Doursat, R., Sayama, H. & Michel, O., eds. (2012). Morphogenetic Engineering: Toward Programmable Complex Systems. "Understanding Complex Systems" Series, Springer-Verlag.

- Doursat, R., Sayama, H. and Michel, O. (2013). A review of morphogenetic engineering. "Frontiers of Natural Computing" (FNC 2012) Special Issue. Lones, M., Tyrrell, A., Stepney, S. & Caves, L., eds. Natural Computing 12(2): 517-535.
- Fernández, J. D., Doursat, R. & Vico, F. J. (2012) The evolution of controller-free molecular motors from spatial constraints. Proc. of 5th Spatial Computing Workshop (SCW 2012), in 11th Int'l Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2012), vol. W21, June 4-8, 2012, Valencia, Spain: pp. 19-24. IFAMAAS.
- Fleischaker, G. R. editor (1992). *Autopoiesis A Debate: Controversy over Physical, Biological and Social Systems*. International Journal of General Systems, Volume 21.
- Hutton, T. J. (2007). Evolvable Self-Reproducing Cells in a twodimensional artificial chemistry, Artificial Life, 13(1):11-30.
- Langton, C. G. 1984. Self-reproduction in cellular automata. *Physica D* 10:135–144.
- Luhman, N. (1986). The Autopoiesis of Social Systems, Sociocybernetic paradoxes, pp. 172-192.
- Maturana, H. R. and Varela, F. J. (1973). Autopoiesis: The Organization of the Living, in *Autopoiesis and Cognition: The Realization of the Living* (Maturana and Varela 1980), pp. 59-138. Dated 1973. First published 1972 in Chile under the title *De Maquinas y Seres Vivos*, Editorial Universitaria S.A.
- Maturana, H. R. and Varela, F. J. (1980), Autopoiesis and Cognition: The Realization of the Living, Vol. 42 of Boston Studies in the Philosophy of Science, D. Reidel Publishing Company, Dordrecht, Holland. With a preface to 'Autopoiesis' by Stafford Beer. Series editors: Robert S. Cohen and Marx W. Wartofsky.
- McMullin, B. (2004). 30 Years of Computational Autopoiesis: A Review, ALife, 10(3): 277-295.
- McMullin, B. and Groß, D. (2001). Towards the implementation of evolving autopoietic artificial agents, in *Advances in Artificial Life, Procedings of the 6th European Conference on Artificial Life (ECAL2001)*, Kelemen, J. and Sosík, P. editors, pp. 440-443.
- McMullin, B. and Varela, F. J. (1997). Rediscovering Computational Autopoiesis, ECAL-97 Presentation
- Ono, N. and Ikegami, T. (1999). Model of self-replicating cell capable of self-maintenance, in *Proceedings of the 5th European Conference on Artificial Life (ECAL1999)*, Floreano, D., Nicoud, J.-D. and Mondada, F. editors, pp. 399-406.
- Ono, N. and Ikegami, T. (2000). Self-maintenance and self-reproduction in an abstract cell model, *J. of Theoretical Biology*, 206:243-253.
- Ono, N. and Ikegami, T. (2001). Artificial chemistry: Computational studies on the emergence of self-reproducing units, in *Advances in Artificial Life, Proc. of the 6th European Conference on Artificial Life (ECAL2001)*, Kelemen, J. and Sosík, P. editors, pp. 186-195.
- Pelaez, N. (2009). Autopoiesis Model, EECS 372/472 Course at North-western University. http://ccl.northwestern.edu/courses/mam2009/student_work/ Autopoiesis.html.
- Rasmussen, S., Chen, L., Deamer, D., Krakauer, D. C., Packard, N. H., Stadler, P. F., & Bedau, M. A. (2004). Transitions from nonliving to living matter. *Science*, 303(5660), 963-965.
- Sayama, H. (2000). Self-replicating worms that increase structural complexity through gene transmission, *Artificial Life VII: Proc. of the 7th Int'l Conference on Artificial Life*, M. A. Bedau, J. S. McCaskill, N. H. Packard and S. Rasmussen, eds., pp.21-30, MIT Press.
- Sayama, H. (2009). Swarm Chemistry, Artificial Life, 15(1):105-114.
- Sirmai, J. (2011). A schematic Representation of Autopoiesis Using a new kind of Discrete Spatial Automaton, 11th European Conference on Artificial Life (ECAL2011)
- Sirmai, J. (2013). Autopoiesis facilitates Self-Reproduction, 12th European Conference on Artificial Life (ECAL2013)
- Swenson, R. (1992). Galileo, Babel and Autopoiesis (It's turtles all the way down), *International Journal of General Systems*, 21:207-228.
- Varela, F. J., Maturana, H. R. and Uribe, R. (1974). Autopoiesis: The Organization of Living Systems, its Characterization and a Model, BioSystems, 5:187-196.
- Wilensky, U. (1999). NetLogo, ccl.northwestern.edu/netlogo, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
- Zeleny, M. (1977). Self-Organization of Living Systems: A formal Model of Autopoiesis, *International Journal of General Systems*, 4:13-28.

Applying Self-Assembly and Self-Reconfigurable Systems for Printer

Masayoshi Mitsui¹, Atsushi Masumori¹, Ryo Asakura¹ and Hiroya Tanaka¹

¹Keio University mitsuimasayoshi@gmail.com

Abstract

We demonstrate the application of self-assembly system for human manufacturing processes at meso- and macroscale in order to pursue and expand the engineering applicability. First, we implemented a prototype referred to as self-assembly printer which can generate a dynamic, transformable, two-dimensional molding and reuse the units that compose the printed objects. Secondly, noteworthy technological components to explain our prototype system is introduced in this paper. Lastly, some potential application based on the system such as "Self-Assembly Electronic Circuit (SAEC)" is implemented and proposed. As a result of these researches, our system attains scalability and alleviates the need for complex and accurate movements of assembler in traditional manufacturing system.

Introduction

Human manufacturing systems typically require high complexity that can only be achieved by top-down or laborious manual assembly. As technologists pursue smaller and more complex structures, traditional manufacturing schemes will not be able to meet such demands, because they are limited in their scalability, robustness, and complexity. As a solution to this problem, applying Self-assembly to human manufacturing process has been a long sought goal in many fields of both academia and industry (Griffith (2004)). Self-assembly has been used to create structures at the nano- and microscales using techniques such as chemical bounding, geometric interactions, and magnetic field. Even at the meso- and macro-scales, many techniques and basic theories to design self-assembling and self-reconfigurable systems has been proposed and implemented in the field of robotics (Cheung et al. (2011)). Although these robotic systems are impressive and are approaching functionality, they offer little hope in terms of scalability for large applications or complex structures, because of its high application cost, failure of electronics and miscommunication between machines. Therefore, this paper focuses on physical properties and geometric forms of units and implements Self-Assembly Printer which applies self-assembly system to manufacturing processes. In detail, we implemented simple designed units and an assembler. And then we constructed a mathematical model based on this designed units. Consequently, self-assembly printer system was proposed and implemented. Moreover, some potential applications based on the system such as SAEC (Self-Assembly Electronic Circuit) will be proposed here.

Self-assembly printer

Self-assembly printer system consists of 22 types of minimum units, a mathematical model, and a simple designed assembler. First, an output shape at seven-segment display is determined through a graphical user interface. Second, a programmed software analyses the given shape in order to find the most appropriate Hamiltonian path and inform the assembler of the accurate order of units. Third, the assembler rotates to bring the units to the front of the extruder and pushes them out through it. By completing these steps, the self-assemble printer system can print some character or number at seven-segment display (see figure.1).

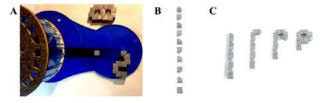


Figure 1: "(A) an assembler which rotates and allocates units. (B) a line of units allocated to form character "p". (C) an example process of character "p" being printed as a seven-segment display on self-assemble printer system".

Design of minimum units

In this part, we will introduce a prototyped unit and a way of infusing binary codes into them as a genetic code. First, we focused on the geometric forms and the direction of magnetic flux embedded within the units, and associated the physical properties with binary codes as seen the figure.2. Second, we prototyped each units which have only four relative patterns (with Objet260: 3D Printer) although we had to fabricate 22 types without considering their geometric symmetry. Third, we introduced catalyst unit which will come

off the printed object in order to achieve the required shape and to assemble units more efficiently (the details are shown below).

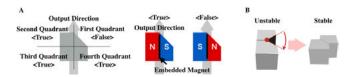


Figure 2: "(A) Minimum unit's code which is compatible with genetic code. (B) Interaction between units".

Bridge-Method

As we understand how any shape is printed with only single path, we confront Hamiltonian path problem. There is an effective mathematical method proposed by (Bachrach et al. (2009)) to find a Hamiltonian path in any two- or threedimensional shape. However, in the method, the number of vertices of given the graph are forced to be quadrupled in two-dimension case and thus the scale of the object becomes bigger. Therefore, we developed new method called "Bridge-Method" to create a Hamiltonian path with catalyst units. First, our method searches all odd points in a given graph not serially but parallel. Second, our method determines two points and bridges them with catalyst units in order to make an Euler path. If there are more than two odd points, our method would evaluate multiple solutions with an evaluation function. Finally, Bridge-Method is proposed as a more efficient method to make Hamiltonian path to achieve the goal more efficiently for our system.

Experimental Result

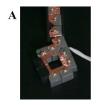
We have implemented self-assembly printer system and demonstrated how it prints various shapes. First our system was able to print all characters and numbers at seven-segment display as seen the below figure.3. However, sometimes our system fails to print the required shape because of influence of friction force. For the same reason the system is unable to print large numbers of units. Second, we introduce "SAEC: Self-Assembly Electronic Circuit" as a potential application of our system in the below part.



Figure 3: "All characters and numbers printed by self-assembly printer".

Self-Assembly Electronic Circuit

We implemented SAEC as a potential application with the self-assembly printer. In detail, SAEC units are designed with copper foils in the same pattern for each units, and an electronic circuit is accomplished just after constructing the object. Most fabrication of microelectronic devices is carried out by photolithography and is unable to reuse or replace its components. SAEC offers a possibility to restore itself to its original condition or extend its function locally. As a future work, we propose "DSAEC: Dynamic SAEC" composed of more intelligent units. Like the SAEC, DSAEC units are all designed in the same pattern, however, they are embedded with different electronic components such as LED, battery, and sensor. We will implement units that are constructed through static self-assembly in our system, and moreover, will become agents and interact with other agents dynamically.



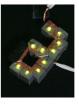




Figure 4: "(A) Self-Assembly Electronic Circuit as a potential application of self-assembly printer. Each units are designed with copper foils in the same pattern. (B) an early prototype of Dynamic SAEC'.

Conclusion

The self-assembly printer system and an example of application was implemented and proposed here. The self-assembly printer system attains scalability to enable the assembler to print an object even if the scale is bigger than the size of the assembler. The system also alleviates the need for complex and accurate movements of assembler compared with top-down human manufacturing system. As our future work, we will develop the self-assembly system which can construct arbitrary three-dimensional forms. And also, we will apply this system for constructing small and complex structures of electronic circuit boards without highly-precise-assemblers.

References

Bachrach, J., Zykov, V., and Griffith, S. (2009). Folding arbitrary 3d shapes with space-filling chain robots: Folded configuration design as 3d hamiltonian path through target solid.

Cheung, K. C., Demaine, E. D., Bachrach, J. R., and Griffith, S. (2011). Programmable assembly with universally foldable strings (moteins). *Robotics, IEEE Transactions on*, 27(4):718–729.

Griffith, S. T. (2004). Growing machines. PhD thesis, Massachusetts Institute of Technology.

Self-organization of Symbiotic Multicellular Structures

Jean Disset¹, Sylvain Cussat-Blanc¹ and Yves Duthen¹

¹University of Toulouse, IRIT - CNRS UMR 5505, 2 rue du Doyen Gabriel Marty, 31042 Toulouse, France jean.disset@irit.fr

Abstract

This paper presents a new model for the development of artificial creatures from a single cell. The model aims at providing a more biologically plausible abstraction of the morphogenesis and the specialization process, which the organogenesis follows. It is built upon three main elements: a cellular physics system that simulates division and intercellular adhesion dynamics, a simplified cell cycle offering to the cells the possibility to select actions such as division, quiescence, differentiation or apoptosis and, finally, a cell specialization mechanism quantifying the ability to perform different functions. An evolved artificial gene regulatory network is employed as a cell controller. As a proof-of-concept, we present two experiments where the morphology of a multicellular organism is guided by cell weaknesses and efficiency at performing different functions under environmental stress.

Introduction

In nature, the cellular specialization (or differentiation) process allowing a single celled organism to grow organs of various shapes and functions is a key mechanism in the development of complex morphologies, behaviors and developmental strategies. Over the past few years, many developmental models have been designed to simulate the growth of virtual multicellular organisms. They have been dealing with different levels of biological realism that directly impact on the complexity and the computational cost of the model. From the cellular automata in the 1960s (Von Neumann et al., 1966) and its offspring (Gardner, 1970; De Garis et al., 1999; Chavoya and Duthen, 2008) to today's models (Hotz, 2004; Joachimczak and Wróbel, 2008; Doursat, 2009; Cussat-Blanc et al., 2012a), this field of research evolved by getting closer to the biological reality. However, though cell differentiation was frequently explored (e.g. French flag problem), the models used evolutionary algorithms guided by an exogenous engineered fitness function describing a predefined morphology.

In the last decades, some cell development models also explored the creation of multicellular organisms which would be evaluated not directly in regard with their morphology but in terms of capability to fulfill a certain function. Co-evolution of morphology and control system showed interesting results, using global control methods such as neural networks in order to, for example, push a block (Bongard and Pfeifer, 2003), or using an artificial gene regulatory network to locally contract cells and swim, such as what Joachimczak et al. did, using a two phase development and explicitly forcing animats to stop growing through fitness function (Joachimczak and Wrobel, 2012) or such as what Schramm did with, however, a subobjective of the evolution being an explicit shape description (Schramm et al., 2011).

The work presented in this paper offers to investigate the emergence of morphologies, developmental capabilities and behaviors of a multicellular simulation. Contrary to many other models (previously cited), this system does not rely on any global controller nor has to enforce any particular differentiation pattern or morphology through a fitness function, but has to find developmental strategies to adapt to and to compose with a hostile environment as well as a more biologically plausible specialization process.

We present a developmental model (named SOMAS, for Self-Organizing Multicellular Artificial Systems) in which cells embed a simplified simulation of a cell cycle regulated by an artificial gene regulatory network. Thus, cells are given the possibility to divide, to specialize, to stay in a quiescent state or to commit apoptosis. Moreover, a massspring-damper system simulates the cell mechanics with intercellular collisions and adhesions, and a simplified artificial chemistry is used for both nutrient transformation into energy and morphogen diffusion. The organisms develop from one stem cell and an evolutionary algorithm is used in order to select the ones that succeed in developing in a hostile environment. One of the main goals of the model is to present a simple but efficient way to bring biological plausibility to the specialization process. Our work is based on the idea that the specialization process of biological cells selects characteristics to develop while other domains are abandoned. The interesting side effect of such process is the need for cellular cooperation through the creation of interdependent functional organs. In our model, we quantify specialization states in terms of weaknesses and efficiency in realizing simple tasks (to transform nutrients into energy, to store energy, to resist to environmental stresses). We thus try to avoid the creation of omnipotent specialization states and instead create a "meaningful" differentiation process which abstracts the underlying physical realities of in-vivo cell specialization. This, with the conjunction of a very simple fitness function which only takes the time of survival into account, could make for the emergence of developmental strategies that are not explicitly described but instead result from the intrinsic strengths and weaknesses of specialization states (which can be dictated by the physical realities of the considered cellular units) together with environmental constraints.

The paper is organized as follows: next section details our model through the morphogens, nutrients and energy management, the cells capacities, physics and controller and the genetic algorithm used to optimize the gene regulatory network of the cells. Third section presents two proof-of-concept experiments that shows the capacity of the model to come with different strategies by only using the surviving duration of the organism as fitness function. Finally, the paper concludes on possible extensions and future experiments that our model is expected to tackle.

Model

SOMAS is designed to provide some bio-inspired capabilities to the cells. In particular, we focused on the following three main components: an *artificial chemistry* providing the cells with the ability to interact and to produce energy, a *cell physics* model that simulates the collision and adhesion mechanisms and a *cell cycle* abstraction with the possibility to *specialize* into different cellular states, controlled by a *gene regulatory network*.

Artificial chemistry

The chemistry is here aimed toward two main goals. First, the diffusion of morphogens in the environment is used by the cells in order to communicate and gain, for example, positional informations. Secondly, it provides the cells with the basics for energy management. Both aspects are detailed in the following paragraphs.

Morphogens Morphogens are signaling molecules produced by cells. They play an essential role in embryogenesis. Often produced from a localized source, the morphogen concentration gradient helps cells to localize themselves and trigger cell specialization. We use a simplified morphogen diffusion system based on a grid in which cells inject and sense morphogens concentrations level. An artificial gene regulatory network described hereafter and embedded in each cell governs the morphogen production rate.

A morphogen is characterized by its attenuation speed (from which can be deduced its range) and its evaporation rate. At each time step, morphogen levels are updated on the morphogen grid: their concentration are first decreased according to their evaporation rate, before being diffused to the neighboring grid cells. When a grid's cell diffuses its concentration to its surrounding, the neighbors receive its morphogen concentration multiplied by the attenuation speed of the morphogen. The morphogen concentration C_i received by a grid cell i from a neighboring source of morphogen with a concentration of C_S is thus equals to:

$$C_i = C_S * (1 - R_m * d(i, S))$$

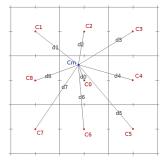
with $R_m \in [0, 1]$ the attenuation speed of morphogen m and d(i, S) the distance between the centers of cells i and S.

A morphogen grid cell concentration always keeps the maximum concentration value offered (which either directly comes from a cell placed on it or from the diffusion of its neighbors grid cells). When a cell senses the morphogen concentration for its position, the morphogen grid computes and returns the average concentration of the 9 nearest grid's cells centers weighted by their distance to said position. The same interpolation process is used when the cell diffuses its morphogens into the grid cells. With grid cells in the same order of size as the cells, this system allows for "smooth enough" morphogens gradient (as can be seen in figure 3) formation without the algorithmic complexity of a purely continuous model. Morphogen concentration C_m at position m is thus equals to:

$$C_m = \frac{\sum_{i=0}^{8} \frac{C_i}{d_i}}{\sum_{i=0}^{8} \frac{1}{d(i,m)}}$$

with C_i the morphogen concentration in grid cell i and d(i,m) the distance between m and cell i center's position. Figure 1 depicts this morphogen concentration computation.

In figures 2 and 3, three morphogens are produced by different cells. Their concentrations are mapped to the cyan,



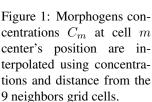




Figure 2: Visualization of morphogens concentrations (3 morphogens: cyan, magenta and yellow) as sensed by cells.

magenta and yellow colors. Figure 2 shows a visualization of the morphogen concentrations inside each cell (only the outer three cells are producing morphogens) and the figure 3 shows the morphogen concentrations on the grid. Note that the morphogen concentration levels are not cumulative: if two cells produce the same amount of a same morphogen in the same place, the morphogen concentration level will be as high as if there was only one cell. This non-cumulative diffusion system presents similarities with Doursat's (Doursat, 2009). It has been proven to be efficient and precise enough for virtual embryogenesis.

Nutrients, ambrosia and energy Cell consume energy to perform different actions: division, apoptosis, quiescence or specialization. Division is, for example, a very costly operation whereas quiescence is a sleeping state where the cell consumes less energy. The energy level is coded as a floating-point number between 0 and 1 in each cell. When a cell's energy level reaches 0, it dies. The average energy consumption per time step being in the order of magnitude of 0.1, pure energy storage is not efficient and, in order to survive, cells must produce energy at a steady pace. However, this energy cannot be shared nor stored.

Therefore, another form of energy is introduced: *ambrosia*. It is a special "molecule" which can be stored and diffused. Ambrosia can be turned into pure usable energy without loss. This molecule is thus to be compared with glucose or lipid in its energy storage purpose. Ambrosia diffusion is done from one cell to its direct neighbors (cells that have direct adhesive contact) with a certain efficiency, defined by the diffusing cell's specialization state.

Again, this ambrosia cannot be produced out of nothing, and this is where nutrients are needed. They are the raw components at the root of the energy chain. Cells can transform nutrients into ambrosia and their efficiency in doing so, which is crucial to them, is also described in the cell's spe-

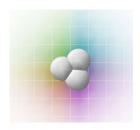


Figure 3: The grid Figure diffuses morphogens forces between its centers. The concentrations are then cells of interpolated to allow for the formation of smooth comparing gradients.



Figure 4: Adhesive forces are simulated with springs that connect cells centers, allowing for the formation of compact cellular clusters of various stiffnesses.

cialization state. Nutrients are available in the environment and their concentrations are encoded on a grid. The resolution of this grid, as well as the nutrients level and nutrient regeneration rate can be changed in order to produce different energy related constraints. These constraints, under the use of genetic algorithm, will add various selective pressure to the competing virtual organisms. They are thus different according to the experiment the model will be used for.

Cell physics

Using a precise mechanical engine (close to biological reality) that simulates soft bodies physics allows us to increase the capabilities of the organisms. However, the computational cost needs to remain acceptable. Therefore, the cells physics are based on a mass-spring-damper system. A cell is represented by its mass, radius, stiffness and center position. Contrary to most of existing models (Doursat, 2009; Joachimczak and Wróbel, 2008), cells are not necessarily clustered in one multicellular organism: the mass-spring-damper system is only used to keep the organism consistent by handling the collisions and the inter-cell adhesions.

Collisions The simulated world in which the cells act allows them to move freely, rotating, colliding and bouncing off each other. When two cells are penetrating each other - the distance between their two centers is smaller than the sum of their radii - a collision spring is created between their centers. Its length is equal to the sum of the two cells' radii and its stiffness is defined as the cells' mean stiffness. Collision springs are destroyed when their length is greater than the sum of the two cells radii they are connected to.

Adhesions Cellular adhesion is crucial for the cohesion of tissues. When two cells get close enough from each other - this activation threshold distance is defined by the specialization state - an adhesion spring is created between the two cell's centers. Its length is defined as being the mean radius of the two cells and its stiffness is the mean adhesion strength, also defined by the specialization state of both cells. Figure 4 shows the adhesive forces that allow for the formation of a compact cluster of cells.

Cell actions

Cells are able to perform four main actions, designed after the possible actions and transformations a real living cell can undergo: division, apoptosis, quiescence and specialization.

The *division* is a cellular action of prime importance, as it is the main mechanism behind organ growth. If the exact replication of the mitosis process is not necessary, it is however important to be able to reproduce the external manifestations of division. In this model, a cell can only begin division if it has enough energy and if the membrane pressure is not too high. The cell then grows to double its size (the collision-spring triggering distance increases). The division axis is chosen according to the compression forces the

cell experiences: it will divide perpendicularly to the most compressed axis. When actual division occurs, a clone cell of initial size is created, and energy and nutrients are split in two equal parts.

The *apoptosis* is the programmed "clean" death of a cell. All of its nutrients are diffused to the neighbors and the cell is deleted from the environment.

When a cell enters *quiescence*, it cannot choose to begin another cellular action for a certain number of decision steps and its energy consumption is reduced.

Cell specialization is one of the key mechanisms behind the formation of organs in living organisms and is also at the heart of this work. Specializing cells progressively modify their properties in order to better fulfill certain functions, thus becoming differentiated cells. This specialization process is here represented by the walk through a tree structure in which every node contains a set of values to be assigned to the cell's properties. Once a cell reaches a certain node, which we call *specialization state*, it cannot go back to a previous state and the cell's attributes values are replaced with the ones described by the new specialization state. Such a state contains information about three main aspects: (1) the physical properties (biomass, size, global stifness and adhesion) of the cell; (2) the controller (the subpart of the artificial gene regulatory network to be used, this mechanism being meant to mimic the way a differentiated cell would only express a subpart of its genome); (3) the skill values of the cells that describe its efficiency in performing various tasks, such as using a certain type of nutrients to produce energy or being able to withstand the impact of harmful particles from the environment. This final concept is crucial to the experiments presented hereafter and will be described with more details later on.

As in nature, these actions, among other internal regulations, are regulated according to an artificial gene regulatory network.

Specialization tree

At the heart of the experiments presented in the next section is the concept of "skills design", or the quantification of efficiency for cells from different specialization states to fulfill certain functions. The general idea, inspired by the observation of living multicellular organisms, is that cells cannot be omnipotent, i.e. they cannot do everything with "full" efficiency. Cell specialization is thus always a matter of getting better at doing something and letting other important functions aside. To survive in a complex environment, the cells then have to cooperate and organize so that they optimally use their capacities.

In our model, we implemented this idea with the repartition of a fixed number of skill "points" to distribute between various functions. In the particular case of the experiment presented in the paper, they are the *ambrosia production*, which is the ability to retrieve nutrients from the environ-

ment and turn them into ambrosia, the *diffusion efficiency*, which quantifies the ability to diffuse ambrosia to nearby cells, the *energy efficiency* (the more developed this competence, the less energy the cell consumes at each time step), the *ambrosia capacity*, which quantifies the ability to store a large amount of ambrosia and the *stress resistance*, which is the ability to withstand aggression from the environment (in the experiments presented hereafter, environmental aggressions are represented by particles of various intensities).

Specialization states are thus crafted by distributing skill points into characteristics. Once the balance in the influence of those skill points is carefully established, specialization trees can be described in both their topology and in the skill points distribution for each of their specialization state. In this work, the trees are hand-crafted but they are ultimately meant to be part of the virtual creature's DNA, subjected to evolution.

Gene regulation

In nature, a gene regulatory network is a network of proteins. The cascading interactions between these proteins control genome expression by the use of external signals collected from protein sensors localized on the membrane (Davidson, 2006). These signals activate or inhibit the transcription of the genes, which then determines the cell's behavior. Here, we use an artificial gene regulatory network which is a simplified computational model inspired by its biological counterpart. This kind of controller has been used in many developmental models of the literature (Bongard and Pfeifer, 2003; Hotz, 2004; Flann et al., 2005; Knabe et al., 2008; Joachimczak and Wróbel, 2008; Doursat, 2009; Cussat-Blanc et al., 2012a) and to control virtual and real robots (Nicolau et al., 2010; Joachimczak and Wróbel, 2010; Cussat-Blanc et al., 2012b).

Dynamics The artificial Gene Regulatory Network (GRN) used in this model is based on Cussat-Blanc's model (Cussat-Blanc et al., 2012b). It is however modified in order to allow for a better precision by using continuous tags. Moreover, all the protein concentrations are clamped between 0 and 1, allowing their absolute concentration values to be directly used without need of comparing them to each other. A gene regulatory network is therefore defined as a set of interacting proteins defined by three tags, coded as floating-point numbers between 0 and 1, and a type. The protein tag describes the proteins themselves. The enhancer tag is compared to the protein tags of other proteins to compute the enhancing matching factor between two proteins. The inhibitor tag is used as the enhancer tag to calculate the inhibiting matching factor between two proteins. Finally, the protein type determines if the protein is an input protein, whose concentration is given by the environment of the GRN and which regulates other proteins but is not regulated, an output protein, whose concentration is used as an output of the network and which is regulated but does not regulate other proteins, or a *regulatory* protein, an internal protein that regulates and is regulated by other proteins.

The dynamics of the GRN are calculated as follows. First, the affinity of a protein a with another protein b is given by the enhancing factor $u_{ab}^+ = |enh_a - id_b|$ and the inhibiting $u_{ab}^- = |inh_a - id_b|$ (where id_x is the protein tag, enh_x is the enhancer tag and inh_x is the inhibitor tag of protein x).

Then, the proteins are compared two by two using the enhancing and the inhibiting matching factors. For each protein of the network, the global enhancing and inhibiting values are given by the following equations:

$$g_{i} = \frac{1}{N} \sum_{j}^{N} c_{j} e^{\beta(u_{ij}^{+} - u_{max}^{+})} ; h_{i} = \frac{1}{N} \sum_{j}^{N} c_{j} e^{\beta(u_{ij}^{-} - u_{max}^{-})}$$

$$\tag{1}$$

where g_i (resp. h_i) is the enhancing (resp. inhibiting) value for a protein i, N is the number of proteins in the network, c_j is the concentration of protein j and u_{max}^+ (resp. u_{max}^-) is the maximum enhancing (resp. inhibiting) matching factor observed. β is a control parameter that set up the "usable" distance between the proteins.

The final modification of protein i's concentration is given by the following differential equation:

$$\frac{dc_i}{dt} = \delta(g_i - h_i) \tag{2}$$

Where δ sets up the speed of reaction of the regulatory network: the higher, the more sudden the transitions in the GRN; the lower, the smoother the transitions.

Cell behavior regulation The inputs proteins are connected to both the surrounding of the cell and its internal state. In order to allow behavior to change according to the specialization state, each of these inputs are duplicated for each possible state but only one set per differentiation state is activated at a time, the GRN thus being fed with the same input data but from different input proteins. These data are the *concentration in each morphogen*, the *membrane pressure*, the *energy level of the cell*, the *ambrosia quantity* divided by the maximum ambrosia storage capacity of the cell and the *nutrients concentration* in the environment.

The outputs of the GRN are the following: one protein per *cell action* (quiescence, apoptosis, division or specialization): the action with the highest output protein concentration is chosen and, in case of specialization, the protein with the highest concentration among all the "specialization" output proteins will determine the state the cell must specialize into; one protein for each *morphogen*, which concentration level directly gives the cells morphogen production; one protein to decide the proportion of *ambrosia to be transformed* into energy, and therefore allows the cell to choose between storing ambrosia or refilling its energy level; one protein that gives the proportion of *ambrosia to be diffused* to nearby

cells; another protein giving the amount of *nutrients* a cell should transform into ambrosia, relatively to the maximum consumption capabilities of the cell (defined by its current type) and limited by the environment resources.

When an action is triggered, the GRN powered decision system is locked until the end of the action (e.g. cells cannot trigger apoptosis while they are dividing). However, all other systems (morphogen production, ambrosia transformation and diffusion, etc.) are maintained and the GRN still controls the corresponding outputs.

Genetic algorithm

This model is meant to be optimized by a genetic algorithm. A standard genetic algorithm is used with a crossover that crosses two GRN protein sets and a mutation operator that can equiprobably randomly modify a tag, add a new regulatory protein or delete a regulatory protein.

As we stated in the introduction of this paper, the aim of this work is also to avoid designing a fitness function that would directly guide the organogenesis process. We want morphological complexity to emerge from the differences in specialization states and the need for cells to survive as long as possible in a hostile environment. Each growing organism is thus only evaluated by its survival time, in simulation steps, the longer the better.

Experiments

As a proof-of-concept and as a first step toward more complex experiments on organogenesis, we developed two environments with two different kinds of constraints. The aim is to show the capacity of the model to produce organized developmental strategies without explicit morphology specification hard-coded in the fitness function. The first experiment presented in this paper consists in studying the specialization and auto-organization capacity of cells attacked by harmful particles. The second one adds complexity by constraining the energy of the environment.

First experiment: surviving aggressions

The first of these two experiments takes place in a hostile environment where dangerous particles are aimed toward the center, which is where the first cell appears. We can draw an analogy between this scenario and radiations that would destroy unprepared cells. In a more general way, these particles are a metaphor of any punctual stress that living multicellular organisms need to face while developing.

Initial conditions For this first experiment, we manually set up a specialization tree with only two states: the *nutritive state* and the *defensive state*. Specialization can only occur from the nutritive state to the defensive state and the first cell of the organism is nutritive. The nutritive state is characterized by its ability to produce energy and ambrosia together with a lack of defense against particles. In direct

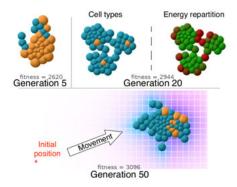


Figure 5: Screen capture of the organism toward the evolutionary process. After 20 generations, the organism organizes nutritive cells (in orange) into a cluster and constantly renews a few defensive cells (in blue) which position are not yet optimal. At generation 20, the organism produces multiple highly organized clusters with a central nutritive cell protected by a field of defensive ones. The energy repartition (showed on the right-hand side, the greener the higher) is remarkably efficient with this strategy. Finally, the best strategy, emerging around generation 40, is movement. Cells organize themselves into a bigger cluster that asymetrically renews cells in order to escape from the center of the environment, where the particles are concentrated, while keeping a heart of nutritive cells surrounded by a shield of defensive ones. The cells use morphogens to position themselves in the colony and drive it in a specific direction.

opposition (and thus complementarity) to the nutritive state is the defensive state, which offers a good defense against particles but a very low nutritive efficiency. Table 1 shows skill points distributions for the relevant skills set.

Property	N	D	Description
Ambrosia	3	0	Nutrients units a cell can turn
Production			into ambrosia at each time step
Ambrosia	10	3	Maximum amount of ambrosia
Storage			that can be stored
Resistance	0	10	Amount of energy that can be
			absorbed from particles before
			the cell dies (here, a particle has
			1 energy)

Table 1: Skill points repartition for the two specialization states N (nutritive) and D (defensive).

Results The results show a global increase in the creatures' life time (the fitness function, expressed in number of frame) over the generations. By observing the best individuals and by looking at the evolution of the selected genome, generations after generations, we can identify three key developmental strategy elements (depicted on figure 5):

- 1. Clusters: After a few generations, cells start to organize themselves in fuzzy clusters where a few nutritive cells are surrounded by protective cells. This strategy seems to be an effective way to use the specialization states: energy providing cells, which are vulnerable, stay protected from the particles by feeding a shell of protective cells. More notably, the formation of these clusters relies extensively on the use of apoptosis, and thus present interesting similarities with the experimental evolution of multicellular "snowflakes clusters" with yeast (Ratcliff et al., 2012).
- 2. Constant renewing: later in the evolution, cells inside the clusters tend to constantly divide and specialize into defensive cells. This makes up for the defensive cells dying because of the aggression of the particles but also because they receive less ambrosia when they are far from the nutritive cells. It is also to be noted that cellular clusters tend to be stable, maintaining an average number of cells throughout their life.
- 3. Shifting from the center. We observe that, over the dozen different evolution runs we made, a special behavior tends to emerge near the "end" of the evolution runs: movement (at cluster level). Organisms tend to avoid the center of the environment which is way more dangerous than the outskirts because of the concentric pattern formed by the particles. They do so by using an unbalanced division ratio and apoptosis: division rate is more intensive on one side of the cluster than on the other side. We verified that this behavior was not just a consequence of the particles being denser in the middle of the environment: the same organisms put in a particle free environment demonstrated the same behavior (depicted by figure 6).

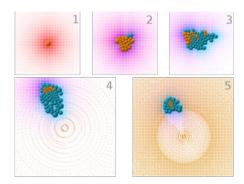


Figure 6: Successive steps of the development of one of the best individuals. Nutritive cells (in orange) surround themselves with a shield of protective cells (in blue) which they feed. Movement toward the outside of the environment can also be noticed (in the direction of the magenta morphogen gradient). Particles become intentionally denser and denser until no organism can possibly survive. This allows for a constant constraint increase throughout the simulation.

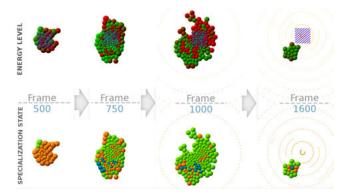


Figure 7: Development of the best organism for the second experiment, showing a 4-steps developmental strategy. Nutritive cells are orange, defensives are green and storage ones are blue. Striped area indicates where nutrients appear. The greener the cells, the higher their energy.

Second experiment: adding an energy constraint

For this second experiment, the cells have to survive under harder constraints. They now not only have to survive aggression from harmful particles, but they also have to compose with a stronger energy constraint. Nutrients are indeed placed on a square centered in the environment where particle density is maximum. Moreover, they are cyclically appearing and disappearing, with an increasing amount of time between every two appearences. Therefore, the organism will have to face episodes of starvation. Moving too soon toward the exterior of the circle in order to escape highly dense particles might thus not be a viable option anymore.

In addition to this constraint, nutritive cells can no longer store large amount of ambrosia. In order for an organism to be able to store it, a third specialization type is introduced: the *storage state*. The analogy could be made with lipid cells which are storing energetic molecules in living beings. The characteristics of this new specialization state are *almost no resistance to particles* (0.1 points), *no nutrients extraction capabilities* (ambrosia production rate is null), *a high ambrosia storage capacity* (10 points) and a *perfect ambrosia diffusion efficiency*. The other two specialization states (nutritive and defensive) are kept identical to experiment 1, except for their ambrosia storage capacity which is now set to 0.5. The first cell is nutritive and can specialize into either defensive or storage.

Results The results show a strong convergence in the development strategies over the 6 evolutionary runs (120 generations of 400 individuals each) launched for this experiment. Figure 7 shows a timelapse of one of the best individuals. The best strategy seems to be the following scheme. First, lots of nutritive cells are created from the initial cell (they can easily survive as the nutrients only disappear for short periods of time at the beginning of the simulation and

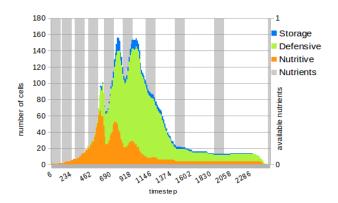


Figure 8: Cell population through time. Grey background indicates nutrients availability. The four steps of the development can be seen: proliferation (frame 0 to 550); shell creation (when particles arrive) and beginning of storage placement (550 to 900); as nutrients are getting rare, creation of a small energy storage cluster, still surrounded with defensive cells linking to the core where nutrients still appear (900 to 1100) and 4) stabilization in a small size cluster (the center of the environment gets abandoned) with a few storage cells spreading all of their ambrosia to the defensives ones.

particles are still far away). Secondly, a shield of protective cells starts to appear. At this stage of the simulation, nutrients are still abundant and the main concern for the organism is to protect them against particles. The organisms that fail in this task in the early generations try to compensate with quick proliferation but eventually die quite early in the simulation due to the particle damages. Then, as nutrient availability starts to become a problem, most organisms start specializing some of their nutritive cells into storage ones. Here, the only difference between organisms that end up with an average fitness and the ones that have the best fitnesses is the localization of these storage cells. The best organisms place the storage cells away from the center of the circle of particles, but with still a network of nutritive cells surrounded by defensives near the center, where nutrients still appear. We can see this behavior between frame 750 and 1000 of figure 7: nutritive cells that were waiting on the extremity of the organism specialize into storage cells in anticipation of starvation. Finally, with the simulation moving forward, appearences of nutrients become extremely rare, and the organisms shrink to a tiny cluster composed of a few storage cells diffusing the ambrosia they collected to a shield of defensive cells. Figure 8 gives an overview of these steps by showing the cell population through time.

The organism was able to find a strategy to adapt to periodical starvation by using the storage state in a suitable way. This illustrates the capacity of the model to step up in complexity in order to manage a new specialization state and harder environmental constraints, as well as how the model can give birth to interesting development strategies.

Conclusion and discussion

This article detailed a model (SOMAS) which aims to provide a base for the organogenesis of virtual organisms. The focus of SOMAS has been made on three main layers: the cellular physics, the simplified cell cycle and the specialization mechanism. The experiments gave encouraging initial results as cells successfully specialized into different available types to better fulfill certain function and adopted a particular morphology and cooperation-based strategy, demonstrating the ability for the model to produce complex developmental plans without any complex fitness design. Morphology and strategy emerged from both the environmental constraints and the intrinsic weaknesses and efficiency the cells had to compose with. The first results showed the formation of clusters and collection of cells that could easily be compared to some of the early living multicellular organisms, which were often barely more than aggregated clusters of specialized monocellular organisms living in symbiosis.

This model aims to bring tools to the artificial life researches for a better abstraction of the consequences and manifestations of living cell specialization. It avoids the design of an omnipotent multipurpose cell as well as the use of complex fitness functions which would explicitly enforce the desired phenotype of the creatures. Our model tries to abstract the underlying complex realities of the cellular specialization process by capturing some of its essential observable implications. One of the idea behind this model is therefore that organogenesis, and thus complexity and diversity in the shape and functions of multicellular organisms, is a direct consequence of the incapacity for a cellular type at being efficient on every front at the same time.

A certain number of enhancements will be at the heart of our future work. First, we presented here experiments where specialization types were manually designed. The next step is to integrate this design directly into the creature's DNA. Optimum cell specialization trees structure as well as skill points distribution will have to be found through an evolutionary process. This means a meticulous balancing work has to be done. We also want to increase the number of attributes specialization states can be formed with.

Another important research path we want to explore is coevolution. Here, we presented experiments where the fitness was kept as simple as possible: it just is the amount of time organisms can survive. Therefore, the model seems well suited for a coevolution approach where all organisms could, for example, compete for their survival in a limited nutrient environment. Using the potentiality of the physical engine by adding physical constraints would also be another way to encourage organisms in finding innovating morphologies.

References

Bongard, J. and Pfeifer, R. (2003). Evolving complete agents using artificial ontogeny. *Morpho-functional machines: The new species (designing embodied intelligence)*, pages 237–258.

- Chavoya, A. and Duthen, Y. (2008). A cell pattern generation model based on an extended artificial regulatory network. *Biosystems*, 94(1):95–101.
- Cussat-Blanc, S., Pascalie, J., Mazac, S., Luga, H., and Duthen, Y. (2012a). A synthesis of the cell2organ developmental model. In *Morphogenetic Engineering*, pages 353–381. Springer.
- Cussat-Blanc, S., Sanchez, S., and Duthen, Y. (2012b). Controlling cooperative and conflicting continuous actions with a gene regulatory network. In *Conference on Computational Intelli*gence in Games (CIG'12). IEEE.
- Davidson, E. H. (2006). The regulatory genome: gene regulatory networks in development and evolution. *Academic Press*.
- De Garis, H. et al. (1999). Artificial embryology and cellular differentiation. *Evolutionary design by computers*, pages 281–295.
- Doursat, R. (2009). Facilitating evolutionary innovation by developmental modularity and variability. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 683–690. ACM.
- Flann, N., Hu, J., Bansal, M., Patel, V., and Podgorski, G. (2005). Biological development of cell patterns: characterizing the space of cell chemistry genetic regulatory networks. *Lecture* notes in computer science, 3630:57.
- Gardner, M. (1970). Mathematical games: The fantastic combinations of john conway's new solitaire game life. *Scientific American*, 223(4):120–123.
- Hotz, P. E. (2004). Asymmetric cell division and its integration with other developmental processes for artificial evolutionary systems. In *Proceedings of the 9th International Conference* on Artificial Life IX, pages 387–392.
- Joachimczak, M. and Wróbel, B. (2008). Evo-devo in silico-a model of a gene network regulating multicellular development in 3d space with artificial physics. In ALIFE, pages 297–304.
- Joachimczak, M. and Wróbel, B. (2010). Evolving Gene Regulatory Networks for Real Time Control of Foraging Behaviours. In *Proceedings of the 12th International Conference on Artificial Life*.
- Joachimczak, M. and Wrobel, B. (2012). Co-evolution of morphology and control of soft-bodied multicellular animats.
- Knabe, J., Schilstra, M., and Nehaniv, C. (2008). Evolution and morphogenesis of differentiated multicellular organisms: autonomously generated diffusion gradients for positional information. Artificial Life XI.
- Nicolau, M., Schoenauer, M., and Banzhaf, W. (2010). Evolving genes to balance a pole. *Genetic Programming*, pages 196– 207.
- Ratcliff, W. C., Denisona, R. F., Borrelloa, M., and Travisanoa, M. (2012). Experimental evolution of multicellularity. *Proceedings of the National Academy of Sciences* 109.5.
- Schramm, L., Jin, Y., and Sendhoff, B. (2011). Emerged coupling of motor control and morphological development in evolution of multi-cellular animats. 5777:27–34.
- Von Neumann, J., Burks, A. W., et al. (1966). Theory of self-reproducing automata.

In-Vitro and In-Vivo Systems

Clonal development is evolutionarily superior to aggregation in wild-collected Saccharomyces cerevisiae

Jennifer T. Pentz¹, Michael Travisano*² and William C. Ratcliff*¹

¹School of Biology, Georgia Institute of Technology, Atlanta, GA 30332, USA ²Ecology, Evolution and Behavior, University of Minnesota, Minneapolis, MN 55108, USA Corresponding author: William Ratcliff, will.ratcliff@biology.gatech.edu. * indicates joint senior authorship.

Abstract

The vast majority of multicellular organisms develop clonally via 'staying together' after mitotic reproduction. Evolutionary theory predicts that cells staying together provides several key advantages over multicellular construction via cells 'coming together', but little empirical work has directly compared these developmental modes. In our previous work evolving multicellularity de novo in the yeast Saccharomyces cerevisiae, cells evolved to form clonal clusters exclusively through postdivision adhesion of mitotically-produced cells, a result that reflects the strong bias towards clonal development in extant multicellular taxa. An equally parsimonious explanation, however, is that cluster development through incomplete cell separation is simply easier to evolve than the production of the adhesive compounds required for aggregation. To disentangle these hypotheses we repeated the experiment of Ratcliff et al (2012), selecting for rapid settling through liquid medium. Instead of using a unicellular ancestor, however, we started our experiment with five wild strains of yeast capable of aggregating into clusters via flocculation. Clonally-developing 'snowflake' yeast evolved and invaded 36/40 experimental populations within 155 transfers, and competition experiments revealed that invading snowflake yeast were substantially more fit than their floc contemporaries. These results support the hypothesis that clonal development is evolutionarily superior to aggregation, and demonstrate that 'snowflake' yeast can readily evolve in diverse, wild-collected yeast strains.

Introduction

The evolution of multicellular organisms from unicellular ancestors is considered a 'major transition' in the history of life on earth. As such, it was one of a few innovations that allowed for the evolution of increased complexity (Smith and Szathmary, 1995). The transition from uni- to multicellularity has occurred at least 25 times in separate lineages (Grosberg and Strathmann, 2007). This transition involved a fundamental shift in biological organization, as individual cells, formally organisms in their own right, evolve to become integral parts of a new, higher-level organism. A key step in the evolution of multicellularity was a transition to larger size, which necessitated the formation of simple cellular clusters (J. T. Bonner, 1998; Boraas, et al., 1998; Kirk, 2005; Pfeiffer and Bonhoeffer, 2003). Selection must then shift from the single cell level to the cluster level, resulting in clusters that are themselves Darwinian individuals (Damuth and Heisler, 1988;

Godfrey-Smith, 2013; Michod, 2005; Smith and Szathmary, 1995).

Construction of an organism from lower-level units that are fully capable of Darwinian evolution is potentially problematic, however, as it may result in evolutionary conflict between the lower- and higher-level units (i.e., cells and multicellular organisms). The potential for conflict is especially strong when selection for multicellular-level functionality results in reproductive altruism among cells (e.g., differentiated somatic cells that are at a reproductive dead-end) (Herron and Michod, 2008; Libby and Rainey, 2013; Michod, 2005). This raises a fundamental question in evolutionary biology: How do the fitness interests of lower and higher-level units become aligned, limiting the negative consequences of evolutionary conflict? Similarly, how are lower-level units (cells) de-Darwinized, limiting the potential for among-cell selection to undermine multicellular-level selection?

Not all multicellular organisms are constructed in the same manner. There are two basic modes of body formation: potentially unrelated cells either 'come together' to form a body, or cells 'stay together' after reproduction, which results in clonal development if the life cycle includes a genetic bottleneck (J. T. Bonner, 1998; Tarnita, et al., 2013). Multicellular organisms have evolved via both routes. For example, the myxobacteria are a group of soil-dwelling bacteria that exhibit a social foraging behavior, coming together to form swarms that increase their feeding efficiency (Olive, 1975; Velicer and Vos, 2009). However, the vast majority of independent transitions to multicellularity have occurred via staying together (J. T. Bonner, 1998), suggesting that this is the superior mode of multicellular development.

Evolutionary theory predicts that multicellular development via cells staying together should provide several key advantages over cells coming together (Grosberg and Strathmann, 2007; Tarnita, et al., 2013). This mode of development limits among-cell genetic variation, especially if the life cycle includes a genetic bottleneck (Grosberg and Strathmann, 1998). Limiting genetic variation among lower-level units has several benefits: 1) It eliminates the potential for evolutionary conflict, since there is little standing genetic d iversity within a higher-level unit for selection to act on (Grosberg and Strathmann, 1998; Hamilton, 1964; Michod, 2005; Michod and Roze, 2001). 2) High among-cell genetic relatedness favors the evolution of traits that increase the fitness of the cluster, even if this reduces the fitness of individual cells.

This facilitates the evolution of cellular division of labor (J. Bonner, 2003; Hamilton, 1964; Queller, 2000). 3) Any genetic variation that arises due to mutation gets partitioned among multicellular offspring, allowing selection to act on the multicellular-level phenotypic effects of *de novo* mutation. This also allows selection to act against mildly deleterious alleles (Grosberg and Strathmann, 2007) that would be hard to select against in chimeric organisms.

Despite the clear predictions of evolutionary theory, it has been difficult to test the hypothesis that multicellular development via staying together should be superior to coming together. This is largely due to a lack of model systems in which both modes of development can be induced. The yeast Saccharomyces cerevisiae can form clusters either by incomplete cell separation after mitosis, producing 'snowflake yeast', or by coming together through adhesive glycoprotein production, a process known as flocculation (Smukalla, et al., 2008). In prior experiments, Ratcliff et al (2012) found that snowflake yeast evolved in 10/10 replicate populations selected for faster settling. This raises the possibility that staying together is superior to coming together in this yeast model system, but it is also possible that this trait simply evolves more readily than flocculation. Here we repeat the experiment of Ratcliff et al., 2012, but rather than starting with a unicellular yeast, we start with wild-collected highly flocculent strains. Our experiment thus 'stacks the deck' in favor of floc, as all yeast start out with the ability to form a cluster via aggregation. If staying together is adaptive, it will need to evolve de novo and invade a population of aggregative yeast.

Methods

Strains, culture conditions, and selection regime

We used five field-isolated flocculating unicellular S. cerevisiae: strains YJM450, YJM454, YPS1000-1, YPS1009-2, and M5-2. Diploids were generated by streaking a and α mating type haploids on YPD agar plates (per liter: 20 g dextrose, 20 g peptone, 10 g yeast extract, 15 g agar). Single strains were isolated through three rounds of single-colony bottlenecking, and diploidy confirmed by tetrad formation after 4 d of shaking incubation at 30°C in sporulation media (per liter: 20 g potassium acetate, 2.2 g yeast extract, 870 mg synthetic amino acid mix, 0.5 g glucose). A single clone of each strain was then used to start eight replicate populations (40 populations total). Yeast were grown in 10 mL liquid YPD in 25×150 mm tubes for 24 h at 30°C, with 250 rpm shaking. Every 24 h, the populations were subjected to settling selection by centrifuging at 100 g for 10 seconds (selection protocol described fully in Ratcliff et al, 2012). Whole populations were cryogenically preserved every 7 d at -80°C.

Constructing fluorescently labeled yeast

Single-strain isolates were obtained from a single population of strain M5-2 (replicate 2) after 60 and 120 days. Each strain was transformed to express either the green fluorescent protein yeGFP or red fluorescent protein dTomato constitutively under the TEF2 promoter, using the LiAc/SS-DNA/PEG method of transformation (Gietz, et al., 1995). Transformed strains were

then imaged with a SPOT Flex 64 MP camera on an Olympus IX 70 microscope at 10 and 20 X magnification.

Measuring yeast phenotypes

The predominant phenotype (e.g., snowflake or floc) was determined microscopically after 7, 28, 60, 91, 120, and 155 days of evolution. After 24 h of growth in liquid YPD (30°C, 250 rpm shaking), 10 μL of culture was placed on a slide under a 25 \times 25 mm cover slip and imaged at 10x and 40x magnification. Snowflake yeast develop by post-division adhesion of cells, as opposed to floc's adhesive aggregation, allowing us to differentiate phenotype via cluster morphology. The numerically dominant phenotype was scored at each time point.

Relative fitness of early snowflake yeast vs. floc yeast

To measure the relative fitness of early snowflake yeast, we isolated a pair of snowflake and floc yeast strains from the 10 populations that contained both phenotypes at transfer 60 (4 from M5-2, 4 from YPS1009-2, 1 from YJM454, and 1 from YPS1000-1, respectively). All strains were grown in liquid YPD for 24 h, then snowflake/floc pairs were diluted 1:200 into 10 mL liquid YPD. These 10 strain pairs were competed over 5 rounds of selection for both growth and settling (100 x g for 10 s). For each pair, three replicate competition tubes were established. Snowflake and floc colonies have a distinct morphology (i.e., smooth colonies for floc and rough colonies for snowflake; see Figure 2a&b). We therefore used plate counts to determine fitness. Each competition tube was plated out at 1:10,000 dilution onto YPD plates after 24 h of growth (pre settling selection), and again after five rounds of growth and settling selection. Colonies were counted on digital images (in ImageJ) made from each plate after two days of growth at 30°C, taken with a Pentax K10D DSLR with a SMC Pentax-D FA 1:2.8 macro lens. Relative fitness was calculated using the ratio of Malthusian growth parameters (Lenski, et al., 1991).

Results and Discussion

Prior experiments selecting for rapid settling of yeast through liquid medium (Koschwanez, et al., 2013; Oud, et al., 2013; Ratcliff, et al., 2012) found that clusters rapidly evolved, but these developed strictly via cells 'staying together' after reproduction, producing the 'snowflake' phenotype. It is unclear, however, if this is because snowflake yeast are actually superior to floc yeast that develop via aggregation, or if snowflake yeast simply arise more readily via mutation. By repeating our prior experiment with five wild-collected highly flocculent *Saccharomyces cerevisiae* strains, we sought to determine if snowflake yeast could invade populations of aggregative yeast. We found that snowflake yeast invaded readily, and in head-to-head competition possessed a substantial fitness advantage over their floc competitors.

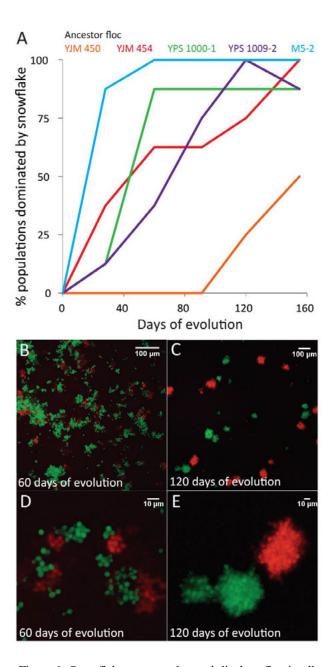


Figure 1. Snowflake yeast evolve and displace floc in all five genetic backgrounds. A) Shown are the percentage of the eight replicate populations of each starting strain that are dominated by snowflake yeast, as determined by cluster morphology. Snowflake yeast develop by cells staying together after reproduction, while floc develop by the coming together of adhesive cells. Morphological differences between clusters are readily apparent in flocculant yeast (B,D) and snowflake yeast (C,E). These yeast were isolated from the same replicate population (M5-2) at either 60 or 120 days of evolution. In each case, a single isolate is labeled with either either the green-fluorescent yeGFP or red-fluorescent dTomato.

Snowflake yeast rapidly appeared in our experimental populations. Within just 28 transfers, snowflake yeast had evolved *de novo*, and risen to high frequency (>50%) in 12/40 experimental populations. Over the course of the experiment (155 days), snowflake yeast evolved in 36/40 populations, generally driving their floc ancestors to extinction (Figure 1a). Indeed, floc regained numerical dominance in just one population taken over by snowflake yeast (Figure 1a).

To measure the relative fitness of invading snowflake yeast, we isolated pairs of floc and snowflake yeast from the 10 populations undergoing invasion at 60 transfers. We grew these pairs in isolation, and then inoculated competition populations with a 50:50 ratio (biomass) of each strain and measured relative fitness over 5 transfers. Invading snowflake yeast were more fit than their floc counterparts in all cases, but this was significant for only 6/10 strain pairs (Figure 2c; $F_{9,40} = 14.543$, P < 0.0001; ANOVA).

So why are snowflake yeast superior to floc yeast? We propose both proximate and ultimate hypotheses. A key aspect of fitness in our experimental system is the ability to rapidly settle to the bottom of the test tube, and for this task snowflake yeast may possess a direct advantage. While floc yeast are certainly able to settle quickly, they rely on a stochastic process for cluster formation. Specifically, clusters are formed through the collision and adhesion of sticky cells, and fragment when shear forces separate cells. In contrast, snowflake yeast form through a far more deterministic process, with daughter cells adhering to their parents after mitotic reproduction. Clusters only fragment when the among-cell tension exceeds cell-cell adhesive strength. Cluster size, and therefore settling speed (Ratcliff, et al., 2013), of individual clusters is heritable and is consistent for different genotypes of snowflake yeast (Ratcliff, et al., 2012; Rebolleda-Gomez, et al., 2012). Alternatively, floc's stochastic aggregative method of cluster formation may result in clusters that are more variable in size and settling speed. As a result, they may lack the ability to consistently produce large, fast settling clusters, allowing invasion by snowflake yeast.

Snowflake yeast may also possess an ultimate advantage, and be more capable of multicellular adaptation. Snowflake yeast develop clonally, while floc yeast form genetically chimeric clusters (Figure 1b,d). Since snowflake yeast clusters are clonal, there is little potential for the evolution of "cheating" lineages that realize a cell-level fitness benefit at the expense of cluster-level fitness (Grosberg and Strathmann, 1998). The chimeric clusters formed by flocculation may result in the evolution of selfish lineages of cells, interfering with multicellular adaptation (Diggle, et al., 2007; Smith and Szathmary, 1995).

Even if cheating is not an issue, the high relatedness among cells in snowflake clusters should favor the evolution of reproductive altruism and cellular division of labor (Hamilton, 1964; Willensdorfer, 2009). The ability for clusters to partition tasks amongst cells may offer many benefits (Smith and Szathmary, 1995; Szathmáry, et al., 2011). Snowflake yeast evolved a simple among-cell division of labor in response to a growth rate-settling rate trade-off. Large clusters grow less quickly than small clusters, due to limited nutrient availability to internal cells. Large snowflake yeast evolved an elevated rate of apoptosis, resulting in the production of proportionally smaller clusters that grow more rapidly (Ratcliff, et al., 2012).

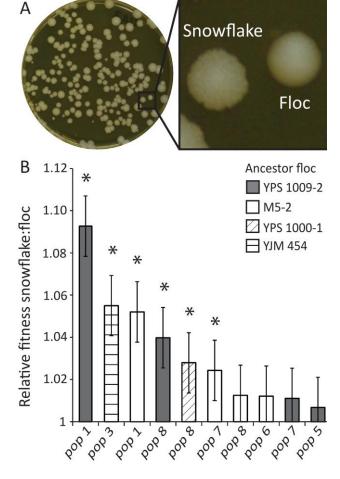


Figure 2. Fitness of early snowflake strains. A) Snowflake and floc colonies have a distinct morphology, allowing us to use plate counts to determine fitness. B) Early snowflake yeast have a significant fitness advantage in 6/10 pairs isolated from the same population at transfer 60. Error bars are the 95% confidence intervals of least-squares means, derived from a 1-way ANOVA. Shown in (A) is a competition plate of a YPS 1009-2 floc-snowflake pair at a 1:10,000 dilution. Asterisks denote significance at the 0.05 level.

Apoptotic cells leave no direct descendants, making this trait costly to individual cells expressing it, but increases the fecundity and fitness of the multicellular cluster (Libby, et al., 2014). This highlights an important shift in the level of selection from the unicellular to multicellular level, which may facilitate the subsequent evolution of multicellular complexity (Damuth and Heisler, 1988; Michod, 2005; Smith and Szathmary, 1995). Genetically chimeric floc clusters are less individuated at the multicellular level, which may limit their capacity for multicellular adaptation (Godfrey-Smith, 2013; Michod, 2005).

Our results raise an obvious question: If snowflake yeast are actually superior to floc yeast, then why are floc more common in nature? We can think of two possible reasons. Flocculation provides protection from environmental stressors (like alcohol

(Hu, et al., 2005) and antibiotics (Lachance, 1990). Unlike snowflake yeast, flocculation can provide a fitness advantage even if opportunities for growth are limited. A rare floc strain can still join a group (and obtain a benefit of stress protection) as long as it produces adhesive glycoproteins (Smukalla, et al., 2008). In contrast, snowflake yeast clusters must grow large in order to gain the benefits of size, which requires a relatively resource-rich environment. Floc yeast should be far better at dispersing, as single cells and small clusters readily break away from a larger group. This both increases the number of propagules formed by a single genotype, and may also increase the distance of dispersal of each propagule. If dispersal is important to fitness, floc yeast may possess a substantial advantage.

Conclusion

'Staying together' and 'coming together' describe the two known modes of multicellular development. Evolutionary theory predicts that staying together is superior to coming together, and here we report the first empirical test directly comparing these two modes of development. Consistent with theory, we find that yeast that form clonal multicellular clusters (snowflake yeast) possess a striking fitness advantage over those that form chimeric aggregates (floc yeast). We hypothesize that the superiority of snowflake yeast could be due to either proximate effects of developmental mode (a larger average cluster size with a smaller variance than floc yeast), and/or ultimate evolutionary effects (more capable of multicellular adaptation than floc yeast). Further experiments are under way testing these hypotheses.

Acknowledgments. The authors would like to thank Adaugo Asouzu for laboratory assistance and Barry Williams for providing the ancestral floc yeast isolates. This work was supported by National Science Foundation grant DEB-1051115.

Author contributions. WR and MT conceived of the project. JP and WR conducted the experiments, analyzed the data, and wrote the paper.

References

Bonner, J. (2003). On the origin of differentiation. *Journal of biosciences*, 28(4), 523-528.

Bonner, J. T. (1998). The origins of multicellularity. *Integrative Biology Issues News and Reviews, 1*(1), 27-36.

Boraas, M. E., Seale, D. B., & Boxhorn, J. E. (1998). Phagotrophy by a flagellate selects for colonial prey: a possible origin of multicellularity. *Evolutionary Ecology*, *12*(2), 153-164.

Damuth, J., & Heisler, I. L. (1988). Alternative formulations of multilevel selection. *Biology and Philosophy*, *3*(4), 407-430.

- Diggle, S. P., Griffin, A. S., Campbell, G. S., & West, S. A. (2007). Cooperation and conflict in quorum-sensing bacterial populations. *Nature*, 450(7168), 411-414.
- Gietz, R. D., Schiestl, R. H., Willems, A. R., & Woods, R. A. (1995). Studies on the transformation of intact yeast cells by the LiAc/SS - DNA/PEG procedure. *Yeast*, 11(4), 355-360.
- Godfrey-Smith, P. (2013). Darwinian individuals. From groups to individuals.
- Grosberg, R. K., & Strathmann, R. R. (1998). One cell, two cell, red cell, blue cell: the persistence of a unicellular stage in multicellular life histories. *Trends in ecology & evolution*, *13*(3), 112-116.
- Grosberg, R. K., & Strathmann, R. R. (2007). The evolution of multicellularity: a minor major transition? *Annual Review of Ecology, Evolution, and Systematics*, 621-654.
- Hamilton, W. D. (1964). The genetical evolution of social behaviour. I. *Journal of theoretical biology*, 7(1), 1-16.
- Herron, M. D., & Michod, R. E. (2008). Evolution of complexity in the volvocine algae: transitions in individuality through Darwin's eye. *Evolution*, 62(2), 436-451.
- Hu, C., Bai, F., & An, L. (2005). Effect of flocculence of a self-flocculating yeast on its tolerance to ethanol and the mechanism. Sheng wu gong cheng xue bao= Chinese journal of biotechnology, 21(1), 123-128.
- Kirk, D. L. (2005). A twelve step program for evolving multicellularity and a division of labor. *BioEssays*, 27(3), 299-310.
- Koschwanez, J. H., Foster, K. R., & Murray, A. W. (2013). Improved use of a public good selects for the evolution of undifferentiated multicellularity. *Elife*, 2.
- Lachance, M.-A. (1990). Yeast selection in nature. *inYeast Strain-Selection*, 21-41.
- Lenski, R. E., Rose, M. R., Simpson, S. C., & Tadler, S. C. (1991). Long-term experimental evolution in Escherichia coli. I. Adaptation and divergence during 2, 000 generations. *American Naturalist*, 138(6), 1315-1341.
- Libby, E., & Rainey, P. B. (2013). A conceptual framework for the evolutionary origins of multicellularity. *Physical biology*, *10*(3), 035001.
- Libby, E., Ratcliff, W., Travisano, M., & Kerr, B. (2014). Geometry shapes evolution of early multicellularity. arXiv preprint arXiv:1403.7556.
- Michod, R. E. (2005). On the transfer of fitness from the cell to the multicellular organism. *Biology and Philosophy*, 20(5), 967-987.
- Michod, R. E., & Roze, D. (2001). Cooperation and conflict in the evolution of multicellularity. *Heredity*, 86(1), 1-7.
- Olive, L. S. (1975). *The Mycetozoans*. New York: Academic Press
- Oud, B., Guadalupe-Medina, V., Nijkamp, J. F., de Ridder, D., Pronk, J. T., van Maris, A. J., & Daran, J.-M. (2013). Genome duplication and mutations in ACE2 cause multicellular, fast-sedimenting phenotypes in evolved

- Saccharomyces cerevisiae. *Proceedings of the National Academy of Sciences*, 110(45), E4223-E4231.
- Pfeiffer, T., & Bonhoeffer, S. (2003). An evolutionary scenario for the transition to undifferentiated multicellularity. *Proceedings of the National Academy of Sciences*, 100(3), 1095-1098.
- Queller, D. C. (2000). Relatedness and the fraternal major transitions. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 355(1403), 1647-1655.
- Ratcliff, W. C., Denison, R. F., Borrello, M., & Travisano, M. (2012). Experimental evolution of multicellularity. *Proceedings of the National Academy of Sciences*, 109(5), 1595-1600.
- Ratcliff, W. C., Pentz, J. T., & Travisano, M. (2013). Tempo and mode of multicellular adaptation in experimentally evolved Saccharomyces cerevisiae. *Evolution*, 67(6), 1573-1581.
- Rebolleda-Gomez, M., Ratcliff, W., & Travisano, M. (2012).

 Adaptation and divergence during experimental evolution of multicellular Saccharomyces cerevisiae.

 Paper presented at the Artificial Life.
- Smith, J. M., & Szathmary, E. (1995). *The major transitions in evolution*: Oxford University Press.
- Smukalla, S., Caldara, M., Pochet, N., Beauvais, A., Guadagnini, S., Yan, C., . . . Latgé, J.-P. (2008). < i> FLO1</i> Is a Variable Green Beard Gene that Drives Biofilm-like Cooperation in Budding Yeast. *Cell*, 135(4), 726-737.
- Szathmáry, E., Calcott, B., & Sterelny, K. (2011). The major transitions in evolution revisited.
- Tarnita, C. E., Taubes, C. H., & Nowak, M. A. (2013). Evolutionary construction by staying together and coming together. *Journal of theoretical biology*, 320, 10-22.
- Velicer, G. J., & Vos, M. (2009). Sociobiology of the myxobacteria. *Annual review of microbiology, 63*, 599-623.
- Willensdorfer, M. (2009). On the evolution of differentiated multicellularity. *Evolution*, 63(2), 306-323.

A novel in vitro metabolic scheme for the construction of a minimal biological cell

Filippo Caschera and Vincent Noireaux

School of Physics and Astronomy, University of Minnesota, 116 Church Street SE, Minneapolis, 55455 Minnesota, United States of America.

filippocaschera@gmail.com; noireaux@umn.edu

Abstract

The bottom-up synthesis of a minimal biological cell is achieved by integrating and connecting three fundamental modules: metabolism, information and self-organization (Noireaux et al., 2011). The execution and connection of these three parts into cell-sized liposomes should generate a system capable of self-reproduction and ultimately evolution. Each of these molecular sub systems is the result of a forward engineering process where bio-parts, e.g. sugars, proteins, phospholipids and nucleotides are interlocked in a functional way (Mann 2008). The creation of a minimal biological cell is certainly one of the most challenging goals of the synthetic biology community (Porcar et al., 2011). The objective of this research area is to elucidate the fundamental design principles found in biology and to understand cellular functions by applying a reductionist approach (Elowitz, 2010). This work also promotes the development of new technologies based on life's principles (Bedau et al. 2010).

The functionalities of biological cells are dependent on the activity of molecular machineries (enzymes). However, such machineries require energy for their physiological function. Therefore, in order to attempt the construction of wet artificial life is essential to reconstitute an in vitro metabolic network that supports the cellular energetic. In biological systems, the energy requirements are stored in the high-energy molecule adenosine triphosphate (ATP). The conversion from ATP to adenosine diphosphate (ADP) is crucial for energy generation. In particular, the hydrolysis of one chemical bond releases inorganic phosphate (iP) and liberates energy used for life processes.

Noireaux's lab has recently developed a unique cell-free transcription-translation (TX-TL) system for synthetic biology applications, as for instance, the possibility of executing DNA programs made of up to ≈ 60 genes (Shin and Noireaux, 2012a, 2012b). In a cell-free system, the rate of protein synthesis depends exponentially on the adenylate energy charge (Atkinson 1968; Matveev et al., 1996). Protein degradation by AAA+ proteases is also highly dependent on the pool of chemical energy available. Therefore, efficient long-lived ATP regeneration and byproducts recycling is at the heart of a minimal cell construction by allowing the execution of larger and larger DNA programs with interesting dynamical behaviors. We will present our recent efforts to design an in vitro metabolism for efficient protein synthesis.

This new metabolic scheme relies on the catabolism of polysugars molecules as energetic resources, and it only exploits the endogenous enzymes present in the cellular extract. Cell-free protein synthesis is improved by addition of maltose or maltodextrin in the reaction mixture. The initial phosphorylation of maltose or maltodextrin produces either

glucose or glucose-6-phosphate, which are intermediates of the glycolysis. In turn, this allows for higher level of sustained ATP concentration through recycling of iP, the byproduct of the transcription/translation processes. We will present biochemical experiments that quantitatively measure several system's parameters: concentration of synthesized protein (a reporter gene eGFP), level of ATP and inorganic phosphate, as well as pH fluctuations during *in vitro* protein synthesis. Recently, we reported the highest protein yield ever achieved with an E. coli cell-free expression system with this new metabolism (Caschera and Noireaux 2013). Therefore, compared to others cell-free expression systems used to design a minimal cell (Ichihashi et al. 2010), it represents a more powerful solution in term of adenylate energy charge.

We are now using this system to develop a minimal cell. One of the current bottlenecks in this research area is the encapsulation of the cell-free TX-TL reaction into cell-sized vesicles of complex phospholipid composition.

Further optimization of the reaction mixture for cell-free protein synthesis, as well as its integration into liposome with an active membrane (Noireaux and Libchaber 2004), could be accelerated exploiting a machine learning approach coupled to robotic workstation for liquid handling (Caschera et al. 2010, 2011).

References

Journal Article

- Atkinson, D.E. (1968). The energy charge of the adenylate pool as a regulatory parameter. Interaction with feedback modifiers. *Biochemistry*, 7:4030-4034.
- Bedau M.A., McCaskill J.S., Packard N.H., and Rasmussen S. (2010). Living technology: exploiting life's principles in technology. *Artificial Life*, 16:89-97.
- Caschera F. and Noireaux V. (2013). Synthesis of 2.3 mg/ml of protein with an all Escherichia coli cell-free transcription—translation system. *Biochimie*, 99:162-168.
- Caschera F., Bedau M.A., Buchanan A., Cawse J., de Lucrezia D., Gazzola G., Hanczyc M. M., and Packard N.H. (2011). Coping with complexity: machine learning optimization of cell-free protein synthesis. *Biotechnology and bioengineering*, 108: 2218-2228.

- Caschera F., Gazzola G., Bedau M.A., Bosch Moreno C., Buchanan A., Cawse J., Packard N., and Hanczyc M.M. (2010). Automated discovery of novel drug formulations using predictive iterated high throughput experimentation. *PloS One* 5:e8546.
- Ichiashi N., Matsuura T., Kita H., Sunami T., Suzuki H. and Yomo T. (2010). Constructiong partial model of cells. *Cold Spring Harbor Perspectives in Biology*, 2(6):a004945.
- Mann S. (2008). Life as a nanoscale phenomenon. *Angewandte Chemie International Edition*, 47:5306-5320.
- Matveev S.V., Vinokurov L.M., Shaloiko L.A., Davies C., Matveeva E.A., and Alakhov Yu B. (1996). Effect of the ATP level on the overall protein biosynthesis rate in a wheat germ cell-free system. *Biochimica Biophysica Acta*, 1293:207-212.
- Noireaux V., Maeda Y.T. and Libchaber A. (2011). Development of an artificial cell, from self-organization to computation and selfreproduction. *Proceedings of the National Academy Science U S A*, 108:3473-3480.
- Noireaux V. and Libchaber A. (2004). A Vesicle Bioreactor As a Step Toward an Artificial Cell Assembly. *Proceedings of the National Academy Science U S A*, 101:12672-12677.
- Porcar M., Danchin A., de Lorenzo V., Dos Santos V.A., Krasnogor N., Rasmussen S., and Moya A. (2011). The ten grand challenges of synthetic life. Systems and Synthethic Biology, 5:1-9.
- Shin J., and Noireaux V. (2012a). An E. coli cell-free expression toolbox: application to synthetic gene circuits and artificial cells. *ACS Synthetic Biology*, 1:29-41.
- Shin J., Jardine P., and Noireaux V. (2012b). Genome replication, synthesis, and assembly of the bacteriophage T7 in a single cell-free reaction, *ACS Synthetic Biology*, 1:408-413.

Magazine Article

Elowitz M. and Lim W.A. (2010). Build life to understand it. *Nature*, 468(7326):889-890.

Towards an Optimal DNA-Templated Molecular Assembler

Jakob L. Andersen ¹, Christoph Flamm ², Martin M. Hanczyc ³, Daniel Merkle ¹ Department of Mathematics and Computer Science, University of Southern Denmark, Denmark

- ² Institute for Theoretical Chemistry, University of Vienna, Austria
- ³ Centre for Integrative Biology CIBIO, University of Trento, Italy

daniel@imada.sdu.dk, jlandersen@imada.sdu.dk
 mhanczyc@gmail.com, xtof@tbi.univie.ac.at

Abstract

In DNA-templated synthesis, reactants are attached to DNA strands and complementary DNA strands are used to control the reaction towards a goal compound. This very general, simple, and still efficient approach has proven to be successful for the design of complex one-pot synthesis for a large variety of compounds. For a given goal compound many different synthesis plans may exist, and all of them can potentially be implemented with many different DNA-templated programs. This raises the issue of how to automatically infer optimal low-level programs based on a high-level synthesis plan or a goal compound only. In this paper we will introduce a computational approach for DNA-templated synthesis based on graph rewriting approaches and the systematic exploration of chemical spaces. We will use them for verification of correctness of real-world synthesis plans as well as to illustrate the non-triviality of finding an optimal DNA assembler program.

Introduction

The "ideal synthesis", according to Wender (Wender and Miller, 1993), may be defined as a simple, safe, environmentally-acceptable, resource-efficient one-step operation, that quantitatively yields the desired complex target molecule from the readily available starting materials. Of course, such an idealized synthesis does not exist for the majority of the synthetically interesting target molecules, however multi-step one-pot reactions approximate Wender's ideal synthesis quite closely.

Therefore considerable effort has been put into the design, development and execution of such one-pot, multistep chemical and biochemical synthesis strategies (Broadwater et al., 2005; Wang et al., 2007; Lundberg et al., 2008; B.Ramachary and Jain, 2011; Denard et al., 2013). This reaction architecture restricts all chemical transformations in the same one-pot space often with sequential addition of reactants or catalysts. Without having to perform multiple discrete purification steps as per normal in synthesis protocols, time is saved as well as resources. If designed properly the one-pot multi step protocol will produce the products of interest while minimizing the undesirable side products common in traditional synthesis.

A design strategy for one-pot synthesis is to covalently attach a short oligomer of DNA to each reactant (Gorska and Winssinger, 2013). In one design, an additional DNA strand is then added to the system as a sequence specific polymeric support that is complementary to both reactant strands, bringing both reactants physically close to one another, see the framed box in the top left of Figure 7. This structured chemical environment increases the effective concentration of the reactants and has been demonstrated to effectively enhance the rate of the reaction towards the desired specific product (e.g., McKee et al., 2010). Several important and distinct chemical reactions can be supported by this design strategy including heteropolymerization, photoligation, click chemistry, condensation, cycloaddition, and many others (see Gorska and Winssinger, 2013).

Recently we described a computational generative chemistry approach based on graph grammar to explore previously intractable complex chemical spaces (Andersen et al., 2013a). As a result we can propose potentially interesting chemical pathways such as synthesis pathways or autocatalytic loops that may be present in the complex mixture. To further define the tools needed to navigate complex chemical spaces, here we present a design strategy based on both one-pot multi-step synthesis and reactions templated by nucleic acid complementarity. We explore how specific chemical pathways may be designed in sequential and parallel steps when reactants are tagged with short oligomeric DNA. We take a computer science perspective on the design of the entire system, by compiling a list of instructions that when implemented in wet chemistry will produce desired outcomes as reaction products.

In the next two sections we will introduce to DNA-templated synthesis and necessary definitions for its formalization. A framework based on the combination of graph rewriting approaches and a strategy framework for chemical space exploration will be presented. In the results section we use the framework in order to verify correctness of a DNA-templated synthesis as presented in (McKee et al., 2010) and we analyze two further syntheses with a focus on optimality. In the last section we conclude and give future directions.

DNA Templated Synthesis

State-of-the-Art

While most of the experimental details and questions for the DNA-templated organic synthesis have been worked out over the last couple of years (for a recent review see Gorska and Winssinger, 2013), computational questions are largely open. However, several computational design tasks in the realm of DNA computation and related areas have been approached in the last decade: (1) the primer selection problem (Pearson et al., 1996) which seeks a minimal set of short DNA sequences which specifically bind to a target DNA strand with minimal cross-reactivity, an \mathcal{NP} hard problem, (2) thermodynamic design of multi-stable nucleic acid molecules (Höner zu Siederdissen et al., 2013), (3) sequence design for ensembles of interacting nucleic acid molecules (Zadeh et al., 2011), (4) pathway design for the self-assembly of nucleic acid nano-objects (Yin et al., 2008), (5) the design of DNA interaction networks with defined temporal behavior (Baccouchea et al., 2014), (6) a programming language for composable DNA circuits based on strand displacement as the main computational mechanism has been introduced (Phillips and Cardelli, 2009; Cardelli, 2010). However no theoretical efforts have been undertaken so far to clarify how a synthesis plan (i.e., the instructions how a target molecule is constructed from starting materials) can be "compiled" into DNA-templated assembler programs. This leads to the question: does an optimal synthesis plan translate also in an optimal DNA-templated assembler program? This touches the problem of how many DNA tags should be used to optimize a DNA-templated assembler program, and if such a program does or does not create side products. These issues will be investigated in the following sections.

Definitions

Here we introduce necessary definitions for DNA-templated synthesis, please refer to the framed box of Figure 7 for an illustration. A DNA domain is a sequence of nucleotides (Cardelli, 2010). DNA plus (resp. minus) strands with single domains are denoted with lowercase variables a, b, \dots (resp. over-lined lowercase variables \bar{a}, \bar{b}, \ldots). In Figure 7 plus (resp. minus) strands are illustrated as lines with half arrows pointing to the right (resp. left). DNA strands composed of several chemically linked domains are denoted as a sequence of the corresponding DNA domain variables, where the sequence of domains is always given from 5'-end to the 3'-end, e.g., ab denotes a plus strand of the sequence of the two DNA domains a and b (the blue-red strand in Figure 7), while \overline{ba} denotes a sequence of two domains on the complementary minus strand. A sequence of domains and its complement form a fully stacked helix, e.g., ab and \overline{ba} form a helix. Non-covalent sequences of domains are indicated by a | sign between the domains (e.g., $\overline{b}|\overline{a}$). In this paper an instruction strand as well as a release strand is a sequence of two domains. Without loss of generality instruction (resp. release) strands are always plus (resp. minus) strands. Compounds, i.e., reactants and products, are denoted with uppercase variables A, B, \ldots Compounds can be attached to the 3'-end or 5'-end of a domain. Without loss of generality only minus strands are modified (i.e., only minus strands are DNA adapters for compounds). A domain (or DNA adapter, or DNA tag) modified with a compound (or more specifically a reactant, resp. product) will be called compound-DNA (or more specifically reactant-DNA, resp. product-DNA). If a modification of a domain is at its 5'-end, we will use a superscript prefix notation (e.g., $A\bar{a}$ denotes a DNA strand \overline{a} which has been modified at its 5'-end with compound A). If a modification is at its 3'-end we will use a superscript postfix notation (e.g., \bar{a}^A denotes a DNA strand \overline{a} which has been modified at its 3' end with a compound A).

A complex of instruction strand and one or two compound-DNAs is formally denoted as a pair, e.g., $(ba, \overline{a}^A|^B\overline{b})$ refers to an instruction strand ba which has the two compound-DNAs \overline{a}^A and $^B\overline{b}$ bound to the instruction strand. Note that in this example the compounds A and B are in close vicinity. We conveniently use the special symbol ϵ for the empty strand or an empty molecule. I.e., the complex $(ba, \overline{a}^A|\epsilon)$ has an unbound domain b and the complex $(ba, \overline{a}^A|\epsilon)$ has a domain \overline{b} attached to b and \overline{b} has no compound attached. Note, that ${}^{\epsilon}\overline{b}$ refers to the same compound as \overline{b}^{ϵ} .

An example of how we depict reactions and compound-DNAs is given in the framed area of Figure 7: given the two reactant-DNAs $(\bar{b}^B$ and ${}^A\bar{a})$ and the instruction strand ba, a complex $(ba, \bar{a}^A|B\bar{b})$ is formed via a reaction (i.e., a hyperedge) which is illustrated as rectangle.

Graph Grammars, Generative Chemistries, and Strategies

While several methods for reasoning about chemistry (Dittrich et al., 2001) have been studied, we promote graph grammar approaches (Benkö et al., 2003; Rozenberg and Ehrig, 1997) as the core formal framework to handle chemical transformations. Graph grammars naturally capture the algebraic nature of chemical reactions where molecular graphs operate upon each other to produce (potentially) novel molecular graphs. Molecules are abstracted to edge and vertex labeled graphs while reactions are expressed as graph rewrite rules between input and output molecular graphs. The so-called Double Pushout (DPO) (Rozenberg and Ehrig, 1997) approach provides the most intuitive direct encoding of chemical reactions and the closest connection to the language of chemistry. A DPO transformation rule $p = (L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$ consists of three graphs L, R and K known as the left, right and context graph, respectively, and two graph morphisms l and r that determine how the context is embedded in the left and the right graph. The rule p can be applied to a graph G if the left graph L can be found in G and some additional consistency conditions are satisfied.

Graph grammars can be thought of as context-sensitive grammars, with strings replaced by labeled graphs. Repeated application of a set of reaction rules to a set of chemical units generates the "language" of all possible chemical units reachable from the initial setup. Chemical units can refer to molecules, DNA strands, complexes of instruction strand and compound-DNA, or a combination of such structures. The modeling of DNA strands and compound DNAs as undirected node- and edge-labeled graphs and the modeling of operations on these chemical units via DPO is straightforward and will be shown based on an example. In Figure 1(a) a graph grammar production rule $p: (L \to R)$ is illustrated (for simplicity we omit the context graph K). The diagram shows the labeled subgraphs L (two components) and R (one component), the host graphs G and H, the subgraph matching morphisms (downward arrows), and the production morphism (right arrows) for a chemical operation. In this case a graph grammar rule in order to bind compound-DNA \bar{a}^A to an instruction strand ba is shown, denoted as $G \stackrel{p}{\Rightarrow} H$. Note, that the production could be applied to any other instruction strand, too, as long as L is found as subgraph in the host graph G. Figure 1(b) shows the semantically identical but prettified version, which will be used throughout the paper.

The chemical units and their producing reactions are most conveniently organized in a hypergraph, i.e., the chemical space. While graph theoretical methods are well established for prediction of chemo-physical properties of individual molecules (Gramatica, 2007; Le et al., 2012), analysis of the underlying hypergraph, i.e., the chemical space is, with a few exceptions, lacking. In (Andersen et al., 2012), the NP-completeness of maximizing the production of a desired compound in a given chemical reaction network was proven. The mathematical definition of functional sub-networks, such as a synthesis pathway as flow problems on hypergraphs, allows to detect relevant chemical transformation motifs, e.g., potential synthesis plans in a chemical space.

Recently a generic approach for composing graph grammar rules to define chemically useful rule compositions was introduced (Andersen et al., 2013b). The idea of rule composition has been utilized to define an efficient framework for defining strategies to systematically explore chemical spaces (Andersen et al., 2014). An analysis of the complex chemistry of hydrogen cyanide was recently published in (Andersen et al., 2013a), which uses this framework. Here, we will define high-level constructs for DNA strand computing based on this strategy framework, and we therefore briefly describe a simplified version of it.

Exploration of a chemical space proceeds step-wise. The state of an exploration is a set of molecular graphs (including product-DNAs, complexes, and instruction strands in the case of DNA-templated computing). *Exploration strategies*

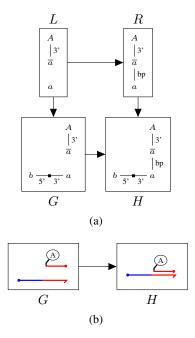


Figure 1: Graph Grammar rule that illustrates a production rule p for binding of a compound-DNA \overline{a}^A to an instruction strand that is composed of the plus strand a and other strands. The edge labels "3'" and "5'" are used to encode directionality of the DNA strands, and the label "bp" encodes the base-pairing of two strands. Applying the production rule p to graph G produces H, denoted as $G \stackrel{\mathbb{P}}{\Rightarrow} H$.

are functions from and to such states. A strategy is defined by a set P of transformation rules in the DPO formalism. The *core strategy* will apply all the rules to all combinations of graphs in the input state. The resulting state consists of all new graphs derived and all graphs that has not been an educt in any rule application. Formally, for an input state F the resulting state F' is given by

$$F' = \{ h \in H \mid \forall G \subseteq F, p \in P : G \stackrel{p}{\Rightarrow} H \}$$

$$\cup \{ g \in F \mid \forall G \subseteq F, g \in G, p \in P : G \not\stackrel{p}{\Rightarrow} H \}$$

As shorthand we write this as $F' = P(F).^1$ To support the dynamic addition of chemical units as needed in DNA-templated synthesis, an *addition strategy* is used, which, given a state F, adds a molecule graph g and yields the state F', i.e., $F' = F \cup \{g\}$. We write this as $F' = \mathtt{add}[g](F)$. The last strategy we will use for DNA strand computation is the *composition strategy*, which corresponds to function composition; given strategies Q_1 and Q_2 their composition $Q_2 \circ Q_1$ is a new strategy. To increase left-to-right readability for large composition chains we opt to write it as $Q_1 \to Q_2$. Instructions of a DNA program as defined in the next section will be based on this formalism.

¹In relation to (Andersen et al., 2014), this is an abstraction of the strategy revive[parallel[P]].

Computational Framework for DNA Templated Synthesis

In this section we will combine the graph grammar approaches for DNA-templated synthesis and the strategy framework in order to define a computational framework for artificial DNA-templated synthesis. The overall structure of the framework is as follows. Based on graph rewriting approaches a chemical space is iteratively expanded (without any DNA tagging), formally this leads to a hypergraph (also called derivation graph DG). Note, that for the artificial DNA-templated synthesis this derivation graph might also be given beforehand. The nodes in DG correspond to chemical compounds and the directed hyperedges correspond to chemical reactions. Within the chemical space DG we infer a pathway that corresponds to a synthesis plan that we are aiming to realize based on DNA-templated synthesis.

To the best of our knowledge, graph rewriting approaches for DNA-templated synthesis do not exist. We use the same graph rewriting framework to augment the chemical compounds of a chemical space with DNA adapters and iteratively infer compounds (now including DNA tagging, cmp. previous section) controlled by a given DNA-templated (synthesis) assembler program (or just DNA program). Besides the derivation graph DG we implicitly assume that the mapping of the tags for a DNA-templated reaction is given, i.e., for each tagged version of a reaction $A+B \rightarrow C+D$ in DG the information is known, if the DNA strand attached to A (resp. B) will be attached to C or D on the product side.

An advantage of using the same framework for expanding the chemical space of compound-DNAs, DNA plus and minus strands, instruction strands, and the corresponding complexes is that this allows for a straightforward and easy integration of our existing graph rewriting approaches. Consequently this means that, e.g., an atom-to-atom mapping is known for all reactions involved. Furthermore, identical methods for pathway inference on the different or integrated chemical spaces could be used. In order to "execute" a given DNA-templated assembler program we will make heavy use of a simplified version of our strategy framework as presented in (Andersen et al., 2014) and outlined in the previous section. If the products after termination of the DNA program do not correspond to the products wished, then we have an unwanted side-effect, i.e., side products we were not aiming for.

Programs for DNA-templated synthesis

DNA programs will be defined in Python², which is used as an interface language to a C++ library in which all the presented approaches are implemented. In our model a program is an exploration strategy, and an empty program is the identity strategy. Each operation in the program adds a spe-

cific strategy by composition. Let the current program be the strategy Q, then the following describes the valid operations.

monomer (compound, DNA-tag, modific. end):
 A compound-DNA corresponding to the arguments of the operation is added to the exploration state, e.g.,

```
monomer("A", tag="\overline{a}", end="5")
```

Formally this corresponds to the strategy composition $Q:=Q\to \mathbf{add}^{[A}\overline{a}].$

• instruction(strand1, strand2): Semantically this means, that an instruction strand is added to the current state, e.g.,

```
instruction("b", "a")
```

Formally this means $Q := Q \rightarrow \mathbf{add}[ba]$.

release (strand1, strand2):
 A release strand is added to the current state, e.g.,

```
release("\overline{a}", "\overline{b}")
```

Formally this means $Q:=Q o \operatorname{add}[\overline{ab}]$

These operations are grouped into sets of add operators, e.g.,

```
add(monomer("A", tag="\overline{a}", end="5"),
   instruction("b", "a"))
```

where each add operation implicitly ends with a reaction strategy $Q_{\rm react}$. This strategy implements the details of rule application for strand displacement, strand binding, and reactions between monomers from the original chemistry. E.g., the add operation above means $Q:=Q\to {\bf add}[^A\bar{a}]\to {\bf add}[ba]\to Q_{\rm react}$. Running a DNA-program corresponds to evaluating the strategy Q on the empty state. The resulting state will be used for DNA program verification, i.e., if unwanted products are in the final state, then the program is not side-effect free.

Results

We present three different setups for DNA program verification. The first example is based on an artificial chemical space and an artificial synthesis plan with four reactants. The second example is a real-world example that illustrates the correctness of the wet-lab results presented in (McKee et al., 2010). The third non-trivial example illustrates how even simple chemical spaces might lead to complicated DNA programs if the program should fulfill a certain optimality criterion.

Synthesis 1: Four Reactants and Four Adapters

In this example we aim at synthesizing product X based on four given reactants A, B, C, and D. A chemical space as depicted in Figure 2 is assumed. In this case the only possible synthesis plan to be implemented with DNA-templated synthesis for compound X is based on the reactions $A+B \to E, C+D \to F, E+F \to X$. This synthesis plan is highlighted with bold lines in the chemical space in

²overlined strings will be used for convenience

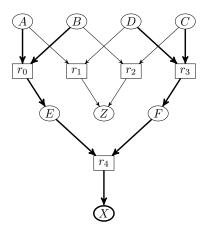


Figure 2: Artificial Chemical Space / Derivation Graph; the synthesis plan to be implemented based on DNA-templated synthesis is highlighted (bold).

Figure 2. We assume that for each reaction in Figure 2 the mapping of the strands from educts to products is given, i.e., (as in our case there is only one product for each reaction), the information is given if for a reaction $A + B \rightarrow C$, the product-DNA of C will be attached to the DNA strand that was attached to A or C. Without loss of generality we assume that the product-DNA of C will inherit the DNA strand attached to A, for reaction $C + D \rightarrow E$ the product-DNA of E will inherit the DNA strand attached to C and for reaction $E + F \rightarrow X$ the product-DNA of X will inherit the DNA strand attached to F. Note, that the chemical space augmented with DNA adapters is much larger; allowing 2 different adapters only leads to a space of 2928 nodes and 8324 reactions even for this toy example, due to the combinatorial explosion of how to build compound-DNAs and complexes of them.

Using four DNA adapters a side-product free one-pot synthesis is straightforward. The monomers A (resp. C) are attached to the 3'-end of the DNA strand \bar{a} (resp. \bar{c}) creating the reactant-DNA \overline{a}^A (resp. \overline{c}^C). The monomers B (resp. D) are attached to the 5'-end of the DNA strands b (resp. d) creating complexes ${}^{B}\bar{b}$ (resp. ${}^{D}\bar{d}$). Using the instruction strand ba, the reactant-DNA \bar{a}^A and $^B\bar{b}$ will form the complex $(ba, \overline{a}^A|^B \overline{b})$. Reactants A and B are now in close proximity and will react, and the complex $(ba, \overline{a}^{\epsilon}|^{E}\overline{b})$ will be formed. At the same when ba is added, the instruction strand dc is added, which in parallel and without interference will form the complex $(dc, \overline{c}^F|^{\epsilon}\overline{d})$. The next step will release the product-DNAs from their instruction strand, which is done by adding DNA-strands \overline{ab} and \overline{cd} . This will release ${}^{E}\overline{b}$ and \overline{c}^F and also form the non-reactive complexes (ba, \overline{ab}) and (dc, \overline{cd}) . Subsequently, a newly added instruction strand dcwill form the complex $(dc, \overline{c}^F|^E \overline{d})$, and E and F will react to X, forming the complex $(dc, \overline{c}^X|^{\epsilon} \overline{d})$. After releasing the reactant-DNA finally the goal product-DNA \bar{c}^X is available and could be easily extracted from this one-pot synthesis plan via the unique tag. Note, that no side products were

```
Input: A Derivation Graph DG
    Output: A Product Set
3
    comp = DNAComputer(dg)
    prog = comp.makeProgram()
    prog.add(monomer("A", tag="a", end="3")
             monomer("B", tag="b", end=
8
             monomer("C", tag="c", end="3"),
9
             monomer("D", tag="d",
             instruction("b",
10
                               "a")
             instruction("d",
11
12
    prog.add(release("a", "b"),
             release("c", "d")
13
14
    prog.add(instruction("b",
    prog.add(release("c", "b"))
15
16
    prog.run()
    return prog.products()
```

Figure 3: DNA-templated program for the synthesis plan (four reactants) given in Fig. 2); four different DNA tags attached to the educts are used.

formed (i.e., all atoms from the educts ended up in the single goal compound X). The overall 4 step DNA program, is shown in Figure 3.

Synthesis 2: Sequence-Controlled Oligomers

In this section we analyze a real-world one-pot multi-step synthesis as presented in (McKee et al., 2010). Using two different mechanisms for the synthesis of oligomers they presented how (i) both mechanisms individually can be used to synthesize specific 4-mers, and (ii) how a specific 6-mer is created by coupling two 3-mers, which have been synthesized in parallel by both mechanisms. In the alternating-strand (resp. same-strand) mechanism the growing strand of the synthesis is transferred (resp. not transferred) between DNA adapters in single steps. More specifically, (i) the alternating-strand mechanism is initiated with a mono-functional ylide monomer (FAM), for which at each step the growing chain is transferred to the incoming monomer. This reaction is stopped by a transfer to a monofunctional aldehyde monomer (BAL); (ii) the same-strand mechanism is initiated with an aldehyde monomer and the growing chain remains on the same DNA adapter throughout the synthesis. The reaction in this case is terminated by addition of an ylide monomer (FAM). For details on determining the tagging of the monomers (all monomers are uniquely tagged), for determining the instruction and release strands, and for the sequence of the addition of the strands we follow precisely the description of (McKee et al., 2010), which we would classify as a straightforward synthesis similar to the artificial example as presented in the first example of the results section. From (McKee et al., 2010) we verified the side-effect free synthesis of the 6-mer as well as the sideeffect free synthesis of the 4-mers, due to space limitations we will focus on the synthesis of a 4-mer.

Chemical Space: The derivation graph based on all possible 2-to-1 reactions of the four monomers has 27 nodes and 47 reactions. The goal compound for the synthesis plan is the

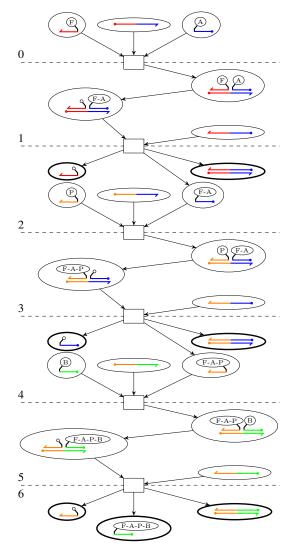


Figure 4: 6-step DNA-templated one-pot synthesis for the oligomer F-A-P-B (cmp. McKee et al., 2010); all the final products are indicated with bold lines, no unwanted side-effect exist.

the oligomer FAM-ALA-PHE-BAL (denoted as F-A-P-B). There are 5 synthesis plans to produce the goal compound, of which the four of them are linear. For the mapping of the tags of educt-DNA to the product-DNA for DNA-templated reactions we straightforwardly follow the descriptions from (McKee et al., 2010).

DNA Program and verification: We omit the source code of the DNA program due to space limitations. However, the program can easily be inferred based on Figure 4, which depicts the automatically inferred sequence of states of the program. The linear synthesis plan is implemented with a 6-step DNA program for templated synthesis. After termination of the program there exist several products. However, the only product-DNA created is the goal compound, therefore the linear synthesis plan is indeed side-effect free (in Figure 4 all products are highlighted with bold lines).

```
Input: A Derivation Graph DG
    Output: A Product Set
3
    prog = comp.makeProgram()
5
    prog.add(monomer("A", tag="a",
6
7
              monomer("B", tag="b", end=
              instruction("b", "a"))
    prog.add(monomer("C", tag="a", end="3"
8
9
              monomer("D", tag="b",
10
              instruction ("b",
11
    prog.add(release("a", "b"))
12
    prog.add(instruction("b",
13
    prog.add(release("a", "b"))
14
15
    return prog.products()
```

Figure 5: DNA program for the synthesis plan given in Figure 2); only two different DNA tags attached to the educts are used; the program has unwanted side-effects.

Synthesis 3: Four Reactants and Two Adapters

Both previous examples used unique DNA adapters for all the educts, i.e., the number of educts was identical to the number of different tags attached to the educts. In order to reduce the number of tags (and therefore increase their diversity) to an optimal amount, the DNA programs need a significant rewrite. In the following example the goal is to find a DNA program for the synthesis plan from Figure 2 with less than four different tags as used in the first example of the results section. It is not hard to see, that it is not possible to use one tag only. For the chemical space and synthesis plan to be implemented, please refer to Figure 2.

Naïve Approach: A DNA program using two tags only is shown in Figure 5. It follows the same idea as the program from Figure 3. However, when the four reactant-DNA \bar{a}^A , ${}^B\bar{b}, \bar{a}^C$, and ${}^D\bar{b}$ are added, the instruction strand ba would lead to side-effect as, e.g., \bar{a}^A and ${}^D\bar{b}$ would form the complex $(ba, \bar{a}^A|^D\bar{b})$, and A and D would react to the unwanted compound Z. While this can be circumvented by sequentially adding instruction strands, the program will still not be side-effect free. After release (line 11) there will be the compound-DNAs ${}^E\bar{b}$ and \bar{a}^F in the pot, but also minus strands \bar{a} and \bar{b} . These will bind to the instruction strand added (line 12) and form, e.g., the complex $(ba, \bar{a}^\epsilon|^E\bar{b})$. The release operation (line 13) will release the goal compound, but in addition also the unwanted side products ${}^E\bar{b}$ and \bar{a}^F .

Correct Approach: A correct program is shown in Figure 6. An illustration of the correctness of the 9-step program is given in the Figure 7. After execution, the only compound-DNA with a non-empty compound attached is the goal compound (step 9 in Figure 7). Similar to classical synthesis this goal is reached by disabling DNA strands that would otherwise indirectly lead to unwanted side products as in the naïve approach. In step 2 of Figure 7, the empty DNA adapter (\bar{a} in the program, depicted red in the figure) that was originally attached to A is disabled by attaching it to an instruction strand (ca in the program, depicted orange-

```
Input: A Derivation Graph DG
     Output: A Product Set
3
     prog = comp.makeProgram()
5
     prog.add(monomer("A", tag="a", end="3"),
                monomer("B", tag="b", end="5"),
6
                instruction("b", "a"))
     prog.add(release("\overline{a}", "\overline{b}"))
8
     prog.add(instruction("b", "c"),
instruction("c", "a"))
10
     prog.add(monomer("C", tag="a", end="3"),
11
12
                monomer("D", tag="\overline{b}", end="5"),
13
                instruction("b", "a"))
     prog.add(release("\overline{a}", "\overline{b}"))
14
     prog.add(instruction("c", "b"))
15
16
     prog.add(release("\overline{b}", "\overline{c}"))
17
     prog.add(instruction("b", "a"))
     prog.add(release("\overline{a}", "\overline{b}"))
19
20
     prog.run()
     return prog.products()
```

Figure 6: DNA program for the synthesis plan given in Figure 2); again, two different DNA tags attached to the educts are used; the program has *no* unwanted side-effects.

red in the figure). Similar in step 5 the adapter originally attached to D (\bar{b} in the program, depicted in blue) is disabled by an instruction strand (cb in the program, orangeblue in the figure). Another important mechanism used is the temporal protection of compound-DNA $^E\bar{b}$ by adding the instruction strand bc (step 2 in Figure 7). This compound-DNA is released again later (step 6 in Figure 7, line 16 in the program) in order to further react.

Conclusion and Future Directions

We introduced an approach for DNA-templated synthesis based on graph grammar approaches. We illustrated important steps towards an organic synthesis compiler which translates the high-level description of an synthesis plan into an executable DNA-templated assembler program. In particular we showed that even the verification of the correctness of a given DNA assembler program is already a non trivial problem. In particular the number of DNA tags has a huge impact on the length and complexity of the resulting DNAtemplated assembler program. Here an expansion of the instruction set via strands which can form hairpin structures depending on the context with reactants hooked up to both ends could probably lessen the number of needed DNA tags. Furthermore, automatic inference of (optimal) DNA assembly program for the synthesis of a specific compound is a major line of our further research. A central question for the future is how optimality in the synthesis plan translates to optimality in the DNA assembler program. It is likely that optimality in these two formulations are competing qualities and therefore the problem must be attacked by a multiobjective optimization approach. Here the fact that our approach is completely formulated in the language of graph grammars is a clear advantage.

Acknowledgements

This work was supported in part by the EU-FET grants RiboNets 323987 and EVOBLISS 611640,, the COST Action CM1304 "Emergence and Evolution of Complex Chemical Systems", and the Danish Council for Independent Research, Natural Sciences.

References

- Andersen, J. L., Andersen, T., Flamm, C., Hanczyc, M. M., Merkle, D., and Stadler, P. F. (2013a). Navigating the chemical space of HCN polymerization and hydrolysis: Guiding graph grammars by mass spectrometry data. *Entropy*, 15(10):4066–4083.
- Andersen, J. L., Flamm, C., Merkle, D., and Stadler, P. F. (2012). Maximizing Output and Recognizing Autocatalysis in Chemical Reaction Networks is NP-Complete. *Journal of Systems Chemistry*, 3(1).
- Andersen, J. L., Flamm, C., Merkle, D., and Stadler, P. F. (2013b). Inferring chemical reaction patterns using rule composition in graph grammars. *Journal of Systems Chemistry*, 4(1):4.
- Andersen, J. L., Flamm, C., Merkle, D., and Stadler, P. F. (2014). Generic strategies for chemical space exploration. *International Journal of Computational Biology and Drug Design*, 7(2/3):225 258. TR: http://arxiv.org/abs/1302.4006.
- Baccouchea, A., Montagnec, K., Padiraca, A., Fujiia, T., and Rondeleza, Y. (2014). Dynamic DNA-toolbox reaction circuits: A walkthrough. *Methods*. doi: 10.1016/j.ymeth.2014.01.015.
- Benkö, G., Flamm, C., and Stadler, P. F. (2003). A graph-based toy model of chemistry. *Journal of Chemical Information and Computer Science*, 43(4):1085 1093.
- B.Ramachary, D. and Jain, S. (2011). Sequential one-pot combination of multi-component and multi-catalysis cascade reactions: an emerging technology in organic synthesis. *Organic & biomolecular chemistry*, 9(5):1277–1300.
- Broadwater, S. J., Roth, S. L., Price, K. E., Kobaslija, M., and McQuade, D. T. (2005). One-pot multi-step synthesis: a challenge spawning innovation. *Org. Biomol. Chem.*, 3(16):2899–2906.
- Cardelli, L. (2010). Two-domain dna strand displacement. In *DCM*, pages 47–61.
- Denard, C. A., Hartwig, J. F., and Zhao, H. (2013). Multistep one-pot reactions combining biocatalysts and chemical catalysts for asymmetric synthesis. ACS Catalysis, 12(3):2856–2864.
- Dittrich, P., Ziegler, J., and Banzhaf, W. (2001). Artificial chemistries a review. *Artifical Life*, 7:225 275.
- Gorska, K. and Winssinger, N. (2013). Reactions templated by nucleic acids: More ways to translate oligonucleotidebased instructions into emerging function. *Angewandte Chemie International Edition*, 52(27):6820–6843.
- Gramatica, P. (2007). Principles of QSAR models validation: internal and external. *QSAR & Combinatorial Science*, 26(5):694–770.
- Höner zu Siederdissen, C., Hammer, S., Abfalter, I., Hofacker, I. L., Flamm, C., and Stadler, P. F. (2013). Computational design of RNAs with complex energy landscapes. *Biopoly-meres*, 99(12):1124–1136.

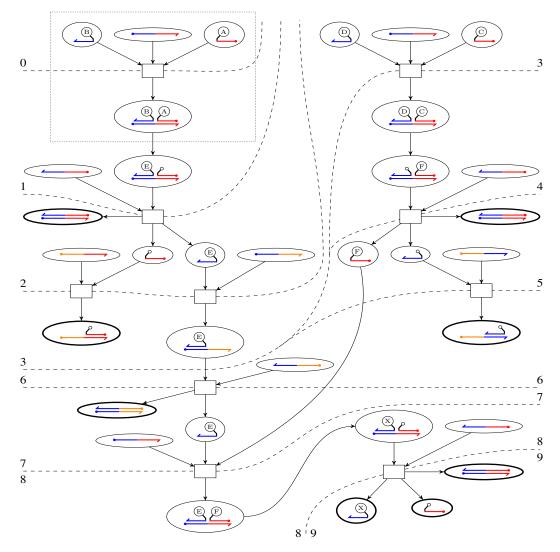


Figure 7: 9-step DNA-templated one-pot synthesis for compound X from Figure 2 using the DNA program from Figure 6; all the final products are indicated with bold lines, no unwanted side-effect exist; dashed lines partition the molecules into the individual states of the mixture between computation/synthesis steps.

- Le, T., Epa, V. C., Burden, F. R., and Winkler, D. A. (2012). Quantitative structure-property relationship modeling of diverse materials properties. *Chem. Rev.*, 112:2889–2919.
- Lundberg, P., Hawker, C. J., Hult, A., and Malkoch, M. (2008). Click assisted one-pot multi-step reactions in polymer science: Accelerated synthetic protocols. *Macromolecular rapid communications*, 29(12–13):998–1015.
- McKee, M., Milnes, P., Stulz, E., Turberfield, A., and O'Reilly, R. (2010). Multi-step DNA templated reactions for the synthesis of functional sequence controlled oligomers. *Angew. Chem. Int. Ed.*, 49:7948–7951.
- Pearson, W. R., Robins, G., Wrege, D. E., and Zhang, T. (1996). On the primer selection problem in polymerase chain reaction experiments. *Discr. Appl. Math.*, 71(1-3):231–246.
- Phillips, A. and Cardelli, L. (2009). A programming language for composable DNA circuits. *J. R. Soc.*, 6(Suppl 4):S419–S436.

- Rozenberg, G. and Ehrig, H. (1997). *Handbook of graph grammars* and computing by graph transformation, volume 1. World Scientific.
- Wang, Y., Ye, X.-S., and Zhang, L.-H. (2007). Oligosaccharide assembly by one-pot multi-step strategy. *Org. Biomol. Chem.*, 5(14):2189–2200.
- Wender, P. A. and Miller, B. L. (1993). Towards the Ideal Synthesis: Connectivity analysis and multibond-forming processes, volume 2 of Organic Synthesis: Theory and Applications, pages 27–66. JAI Press, Greenwich CT.
- Yin, P., Choi, H. M. T., Calvert, C. R., and Pierce, N. A. (2008). Programming biomolecular self-assembly pathways. *Nature*, 451:318–322.
- Zadeh, J. N., Steenberg, C. D., Bois, J. S., Wolfe, B. R., Pierce, M. B., Khan, A. R., Dirks, R. M., and Pierce, N. A. (2011). NUPACK: Analysis and design of nucleic acid systems. *J. Comput. Chem.*, 32(1):170–173.

Computational design of a circular RNA with prion-like behavior

Stefan Badelt¹, Christoph Flamm¹ and Ivo L. Hofacker^{1,2}

¹Institute for Theoretical Chemistry, University of Vienna, Währingerstraße 17/3, A-1090 Vienna, Austria ²Research Group Bioinformatics and Computational Biology, University of Vienna, Währingerstraße 29, A-1090 Vienna, Austria

Abstract

Sophisticated computational methods exist for the conformational design of RNA molecules and they have enabled the design of a multitude of functional RNA devices in the field of synthetic biology. Here we present for the first time an RNA design that mimics the behavior of prions, i.e. sequences capable of interaction triggered auto-catalytic replication of conformations. Our design was computed with the Vienna RNA package and is based on circular RNA that embeds domains amenable to inter-molecular "kissing interactions."

Introduction

During the last decade, the field of Synthetic Biology has impressively illustrated that nucleic acids and in particular RNA molecules are reliable materials for the design and implementation of functional circuits as well as nanoscale devices and objects (Guo, 2010; Khalil and Collins, 2010; Afonin et al., 2013; Ishikawa et al., 2013). The reasons for this success are grounded in the facts that for RNA (1) an experimentally measured energy model exists (Mathews, 2006) (2) regulation at the level of RNA molecules is faster than via the production of proteins and (3) design questions are readily expressed in the discrete framework of binary base-pairing than in continuous interactions between, e.g., the amino acids in proteins.

The "protein only hypothesis" of the scrapie agent (for a review see Aguzzi et al. (2008)) proposes that a prion protein, with an altered (misfolded) β -sheet-rich conformation, starts a catalytic cascade which uses the normally-folded prion proteins as a substrate, converting them to the misfolded form which can self-assemble into fibers. A high activation energy between the normal and the misfolded conformation prevents spontaneous conversion at detectable rates. The formation of a normal-misfolded heteromeric complex may lower the activation energy barrier to convert the normally-folded protein into a misfolded species. This conversion leads to further recruitment of normally-folded proteins in an auto-catalytic process. In essence, a single misfolded prion protein in a population of normally folded ones is enough to convert the whole population via auto-

catalytic structure replication into an all misfolded protein population which self-assembles into long fibers.

This contribution was motivated by the question whether RNA molecules can be designed *in silico* to exhibit the aforementioned prion-like behavior. We show that it is indeed possible to design such an "RNA-prion"; whether the suggested sequence really shows the exponential refolding characteristics awaits experimental verification.

The RNA prion presented here is a 49nt long, circular RNA, designed as a bistable molecule. It thermodynamically favors one structure (S1) if present as a monomer and the other structure (S2) upon increase of its concentration. S2 is designed such that it forms two 10nt hairpins prone to form kissing interactions as reported for the HIV-DIS loop, the genomic RNA dimerization site (Ennifar et al., 2003), a highly conserved stem-loop sequence found in many retroviruses. Importantly, S2 should not only stabilize other S2 conformations at higher concentrations, but actively lower the energy barrier to refold S1 into S2.

We used the program switch.pl (Flamm et al., 2001) of the ViennaRNA package (Lorenz et al., 2011) to design many bistable molecules. On the sequence level we constrained the loop regions of conformation S2 to form a stable kissing interaction. The chosen sequences for interaction have similar free energy to the best kissing interaction examples shown in Weixlbaumer et al. (2004), but differ in point mutations to be compatible with structural constraints for S1. Importantly, in S1, the kissing interaction does not form a regular helix, but the strands are shifted relative to each other (see Figure 2). This asymmetric design enables S2 to open a shorter helical region that has a worse free energy than the subsequent formation of the kissing interaction.

Results

Out of 128 molecules that fulfilled the structural constraints, two showed a very large difference between the energy needed to open a kissing receptor strand and the free energy gain of the duplex interaction with S2. One of these molecules was chosen for detailed analysis and its switch-

ing behavior modeled as described below.

Figure 1 shows the equilibrium between the two structures as a function of RNA concentration. The relative concentrations of monomers [M] and dimers [D] can be computed by the equilibrium partition function. Let Z_M and Z_D be the equilibrium partition functions of the Monomer and two kissing Monomers (Dimer), respectively. We can compute Z_M using the McCaskill Algorithm (McCaskill, 1990) implemented in RNAfold of the Vienna RNA package (Lorenz et al., 2011). Z_D we compute as

$$Z_D = Z_{c1} \cdot Z_{c2} \cdot Z_{dup},\tag{1}$$

with Z_{c1} and Z_{c2} denoting the partition functions of two monomers under the constraint that the first (c1) or second (c2) interaction region is unpaired and thus available for forming a inter-molecular (kissing) interaction. Z_{dup} is the partition function of the inter-molecular duplex formed between the two molecules. This model follows the assumption that dimerization can only involve an interaction between the strands of the kissing interaction.

Since we are interested in the conformations formed upon monomerization and dimerization, we divided the total partition function Z_M into three parts: Z_{S1} , Z_{S2} and Z_o . Z_{S1} and Z_{S2} contain all conformations constrained to form basepairs that can only be formed in structure S1 or S2 respectively, whereas Z_o contains all other conformations, i.e. conformations that are not compatible with both constraints. Constraints are chosen such that (i) the helices formed by S1 and S2 are preserved and (ii) there are no structures fulfilling both constraints. We computed the relative concentration of structure 1 ([S1]) in Monomers and Dimers as:

$$[S1] = \frac{Z_{S1}}{Z_M} \cdot [M] + \left(\frac{Z_{S1+c1}}{Z_{c1}} + \frac{Z_{S1+c2}}{Z_{c2}}\right) \cdot [D]$$
 (2)

where Z_{S1+c1} stands for a partition function that has both the constraint to fold into structure 1 (S1) and the constraint to be unpaired in interaction region 1 (c1). Relative concentrations of S2 were computed accordingly.

Next we computed refolding paths and thus the energy barrier for refolding between S1 and S2, either for a single RNA monomer or an RNA engaged in kissing interaction with another molecule. Due to the short length of the RNA, the problem of finding the best refolding path can be solved exactly, using the program barriers (Flamm et al., 2002).

Figure 2 shows the energy profiles resulting from these paths. Since S1 is the thermodynamically favored state in monomers, we show the refolding path from S2 (-10.70 kcal/mol) to S1 (-12.70 kcal/mol) in the top panel. The barrier of this refolding path is 16.70 kcal/mol, making a non-induced switching of conformations unlikely.

The bottom panel of Figure 2 shows the energy profile for a scenario where an inter-molecular interaction is first formed between one molecule in conformation S1 and a

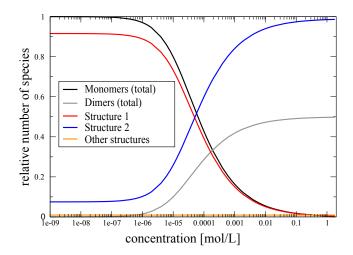


Figure 1: Conformational switching upon change of concentration. The transition from monomer to dimer conformations at around 10nM goes together with a switch from structure S1 to S2.

second in S2, followed by intra-molecular refolding of the first molecule from S1 into S2. S2 is now the favored conformation, since it is stabilized by the kissing interaction. In contrast, S1 is destabilized since one helix cannot be formed together with the inter-molecular duplex. Theoretically, there would be a second possible duplex interaction which requires S1 to open eight base-pairs in two helices, but since this interaction is not thermodynamically favored, it is not depicted in Figure 2.

According to Weixlbaumer et al. (2004), the energy of a 6nt kissing interaction can be computed from the energy of a regular RNA duplex with an additional bonus of -4.20 kcal/mol. For the initiation of the kissing interaction, we calculated the worst-case scenario, where all competing intra-molecular base-pairs of S1 have to open first and then the inter-molecular base-pairs can form. This results in an energy barrier of 6.40 kcal/mol and leads to a new local minimum conformation at -29.70 kcal/mol. For the intra-molecular refolding from S1 to S2, we compute the refolding path given that the kissing-strand is not available for intra-molecular base-pairing, resulting in a barrier of 13.60 kcal/mol. Note that the refolding path of S1 to S2 has to overcome the same barrier conformation as the pathway from S2 to S1, however, the kissing interaction destabilizes S1 and therefore lowers the relative energy barrier. A limitation of the energy evaluation is that the loop entropies of strands involved in the kissing interaction are ignored. The hairipin loop of S2 as well as the interior loop of S1 are evaluated as if they were unpaired and the energy contribution of the duplex (-12.20 kcal/mol) is added.

As an alternative approach, we modeled the kissing interaction as a regular intra-molecular helix. Both molecules are

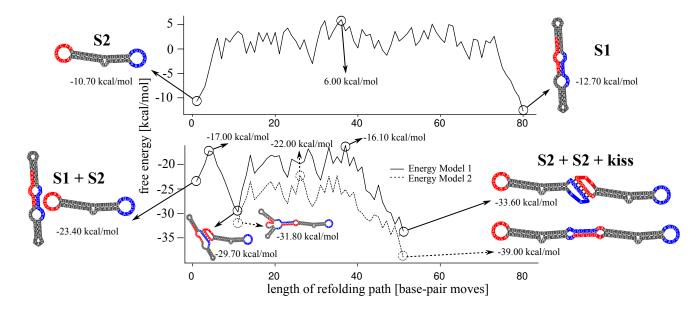


Figure 2: Energy profiles along the refolding path between structure S1 and S2. Top: Refolding of a monomer; bottom: refolding while interacting with a second molecule. Blue and red colored regions are designed to form inter-molecular base-pairing. The lower panel shows a comparison between two energy models that differ in the energy contribution of the loop regions involved in the inter-molecular pairing. In either case, the relative energy of refolding is lower than for the monomer.

cut after the 5'AA of the interacting loops and connected to the beginning of the respective other strand (see Figure 2, structures for Energy Model 2). Hence, we increment the degree of involved loop regions as if a regular helix was formed. The interior loop of S1 becomes a multi-loop and the hairpin loop of S2 becomes an interior loop. Computing the intra-molecular refolding with this altered energy model results in the same path, but with a lower energy-barrier of only $9.80~\rm kcal/mol$.

Conclusion

In this contribution we showed that the computational design of an RNA molecule which shows prion-like behavior is feasible. As in the original prion system, the "misfolded" conformation forms, via a kissing interaction, a hetero-dimeric complex with the native conformation. This interaction destabilizes the native conformation and triggers refolding into the "misfolded" conformation. The calculations show that the kissing interaction lowers the activation energy for refolding drastically. Furthermore, the "misfolded" conformation can oligomerize. Whether the molecule also behaves as expected in the wetlab is currently being checked by our experimental partners.

Acknowledgments

This work was supported in part by the FWF International Programme I670, the DK RNA program FG748004, the EU-FET grant RiboNets 323987, and the COST Action CM1304

"Emergence and Evolution of Complex Chemical Systems".

References

Afonin, K. A., Lindsay, B., and Shapiro, B. A. (2013). Engineered RNA nanodesigns for applications in RNA nanotechnology. *RNA Nanotech*, 1:1–15.

Aguzzi, A., Sigurdson, C., and Heikenwaelder, M. (2008). Molecular mechanisms of prion pathogenesis. *Annu Rev Pathol Mech Dis*, 3:11–40.

Ennifar, E., Paillart, J.-C., Marquet, R., Ehresmann, B., Ehresmann, C., Dumas, P., and Walter, P. (2003). HIV-1 RNA dimerization initiation site is structurally similar to the ribosomal A site and binds aminoglycoside antibiotics. *J Biol Chem*, 278(4):2723–2730.

Flamm, C., Hofacker, I. L., Maurer-Stroh, S., Stadler, P. F., and Zehl, M. (2001). Design of multi-stable RNA molecules. *RNA*, 7:254–265.

Flamm, C., Hofacker, I. L., Stadler, P. F., and Wolfinger, M. T. (2002). Barrier trees of degenerate landscapes. *Z Phys Chem*, 216:155–173.

Guo, P. (2010). The emerging field of RNA nanotechnology. Nature Nanotech, 5:833–842.

Ishikawa, J., Furuta, H., and Ikawa, Y. (2013). RNA tectonics (tectoRNA) for RNA nanostructure design and its application in synthetic biology. *WIREs RNA*, 4:651–664.

Khalil, A. S. and Collins, J. J. (2010). Synthetic biology: applications come of age. *Nature Rev Gen*, 11:367379.

- Lorenz, R., Bernhart, S. H., Höner zu Siederdissen, C., Tafer, H., Flamm, C., Stadler, P. F., and Hofacker, I. L. (2011). Viennarna package 2.0. *Algorithms Mol Biol*, 6:26.
- Mathews, D. H. (2006). Revolutions in RNA secondary structure prediction. *J Mol Biol*, (359):526–532.
- McCaskill, J. S. (1990). The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, 29:1105–1119.
- Weixlbaumer, A., Werner, A., Flamm, C., Westhof, E., and Schröder, R. (2004). Determination of thermodynamic parameters for HIV-1 DIS type loop-loop kissing complexes. *Nucl Acids Res*, 32:5126–5133.

Informational Architecture of the Fission Yeast Cell Cycle Regulatory Network

Hyunju Kim¹, Paul Davies¹ and Sara Imari Walker^{1,2}

Beyond Center for Fundamental Concepts in Science, Arizona State University, Tempe, AZ
School of Earth and Space Exploration, Arizona State University, Tempe, AZ hyunju.kim@asu.edu

Extended Abstract

Introduction

Life seems distinctive in its ability to process and store information. However, precisely what distinguishes information handling in living systems from that of their non-living counterparts remains to be rigorously quantified. A key challenge is that in living systems information is distributed through hierarchically structured networks and appears to play a causal role in the dynamics (Walker and Davies, 2013). Thus the characterization of biological information requires both notions of information flow and of causation. While useful tools for quantifying information transfer and causal structure exist in complex systems research, they have been little applied to distributed information processing in biological networks, particularly at the most fundamental level of biological organization - biochemistry. Thus, which these measures - and in what combination - might best elucidate the informational architecture of biological networks is unknown.

Here we characterize the informational architecture of a Boolean network whose function models the cell cycle of the fission yeast, Schizosaccharomyces Pombe, (Davidich and Bornholdt, 2008) to probe how living systems manage information, how this correlates with dynamics, and how information management may distinguish living from nonliving networks. We also utilize existing frameworks from complex systems research of information dynamics (Lizier et al., 2008) and effective information (Balduzzi and Tononi, 2008). To address how information handling in the biological network manifest in the cell cycle differs from random network architectures, we are particularly interested in how information flows during time evolution of the states of the cell cycle network. We also investigate the information generated via the causal mechanisms of the network and how network topology is related to these characteristics.

Model and Methods

In the fission yeast cell cycle Boolean network, regulatory proteins are modeled as nodes which can have a state of 0 (not present) or 1 (present). The edges model activation or inhibition interactions between two proteins and are

weighted +1 or -1, respectively. The state of a node i, S_i , is updated at each time step with particular rules as outlined by Davidich and Bornholdt (2008).

Local dynamics. To measure the local informational dynamics of the cell cycle network, we obtained time series of the nodes' states for 20 steps, starting from all possible initial states of the network. From these time series, we measured the local transfer entropy $T_{Y \to X}(k)$ defined as

$$T_{Y\to X}(k) = \left\langle \log_2 \frac{p(x_{n+1}|x_n^{(k)}, y_n)}{p(x_{n+1}|x_n^{(k)})} \right\rangle \tag{1}$$

where X and Y are nodes and $\langle . \rangle$ indicates a time average. x_{n+1} is the state of X at time step n+1, y_n is the state of Y at time step n and $x_n^{(k)}$ is k consecutive previous states of X. The transfer entropy encodes how the knowledge about the state of Y can affect prediction of X, given the history of X. If $T_{Y \to X}(k) = 0$, then there is no information flow from Y to X since the information about Y does not affect the prediction of X. If $T_{Y \to X}(k) > 0$, one can say there is informational flow from Y to X.

Global dynamics. The second measurement presented here is the effective information (ei), which characterizes the information generated by the causal structure of a network as whole, when it enters a particular state \mathcal{G} . The effective information for each realized state \mathcal{G} , given by $ei(\mathcal{G})$, is calculated as the relative entropy of the a posteriori repertoire with respect to the a priori repertoire, where the a priori repertoire is defined is as the maximum entropy distribution with all network states equally likely and the a posteriori repertoire is defined as the repertoire of possible states that could have led to the state \mathcal{G} through the causal mechanisms of the system. In other words, ei(G) measures how much the causal mechanism reduces the uncertainty about the possible causes for \mathcal{G} . We calculated the distribution of $ei(\mathcal{G})$ for all states \mathcal{G} which may be caused by the cell cycle. We also calculated the distribution of ei for all possible states in two types of random networks, random networks with global constraint (RNGC) and random networks with local constraint (RNLC). RNGC is a randomized network with the same number of nodes, the same threshold for each node, the same total number of inhibition links and the same total number of activation links as in the cell cycle network. Including the global constraints in RNGC, RNLC is a randomized network with the number of inhibition links and activation links for each node set the same as in the cell cycle. The results for random networks are the averages over 100 random networks of each type.

Results

Fig. 1 shows the transfer entropy, $T_{Y\to X}(k=9)$ for every pair of proteins, Y and X, in the cell cycle network. Nodes labeled in bold correspond to those implicated in a control kernel which sets the attractor of the network's dynamics (Kim et al., 2013). Two groups of nodes, $g_1 = \{PP, Cdc2/Cdc13^*, Cdc2/Cdc13\}$ and $g_2 = \{Ste9, Rum1, Wee1/Milk1, Cdc25\}$, dominate information transfer in the network, and in general $T_{g_1\to g_2}$ is greater than $T_{g_2\to g_1}$.

In Fig. 2, the distribution of ei is shown for all possible effects of the cell cycle network and for the effects of RNGC and RNLC. For each distribution, the area within bars is the number of states that are possible effects for each network. The degree distribution of the cell cycle and RNLC leads to networks with access to more diverse states than RNGC (larger area within the bars). The local constraint leads to similar distributions of ei for the cell cycle and RNLC. However, there are significant differences for large ei where the cell cycle network peaks at ei = 9 and has very few states with ei = 10.

Discussion

The results shown in Fig. 1 suggest that information transfer heavily depends on the local connectivity of the network: the nodes in g_1 have more out-going links, regardless of link type, than any other nodes. Interestingly, the nodes in g_1 , which most provide the most significant sources of information, do not correspond to the control kernel for this network. This suggests that information and causal drivers are differentially distributed within the network. In future studies, we will implement measures of causal information flow to further explore this distinction. For each network in Fig. 2, ei = 10 corresponds to a state with one possible cause and ei = 9 two possible causes. The distribution of ei for the cell cycle therefore suggests that the network is optimized for have multiple possible causes for a particular network state. This is consistent with the attractor dynamics of the cell cycle which exhibits an unusually large basin of attraction, therefore suggesting a novel framework for mapping the information generated by the mechanisms of a network to its dynamics. The results for both measures indicate new connections to be explored mapping the flow and distribution of information to network dynamics which will be explored in future work.

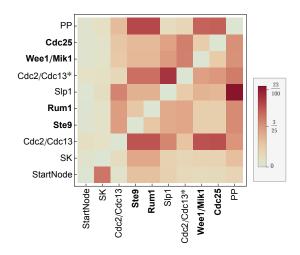


Figure 1: Heatmap for Transfer Entropy (k = 9) of Fission Yeast cell cycle: each cell shows transfer entropy from a node on y-axis to a once on x-axis.

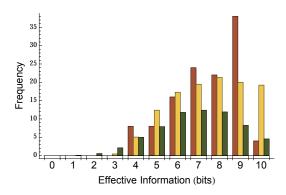


Figure 2: Distribution of Effective Information for Fission Yeast cell cycle (Red), Random Network with Global Constraint (Green) and Random Network with Local Constraint (Yellow).

References

Balduzzi, D. and Tononi, G. (2008). Integrated Information in Discrete Dynamical Systems: Motivation and Theoretical Framework. *PLoS computational biology*, 4(6):e1000091.

Davidich, M. I. and Bornholdt, S. (2008). Boolean Network Model Predicts Cell Cycle Sequence of Fission Yeast. *PloS ONE*, 3(2):e1672.

Kim, J., Park, S. M., and Cho, K. H. (2013). Discovery of a kernel for controlling biomolecular regulatory networks. *Scientific reports*, 3.

Lizier, J. T., Prokopenko, M., and Zomaya, A. Y. (2008). The Information Dynamics of Phase Transitions in Random Boolean Networks. *Artificial Life*, pages 374–381.

Walker, S. and Davies, P. (2013). The algorithmic origins of life. J. Roy. Soc. Interface, 6:20120869.

Evolutionary Art, Philosophy and Entertainment

Modeling and Evaluating the Process of Creating Paintings with Evolutionary Computing

Tomohiro Suzuki and Yasuhiro Suzuki

School of Information and Sciences, Nagoya University ysuzuki@nagoya-u.jp

Abstract

We have modeled the actual process of creating paintings using Evolutionary Computing. In general, an outline and a theme are decided at the beginning of creating a painting, and events are located by the theme, however we evolve initial images to the preselected theme image with this method. We conducted a questionnaire to make the images created by this method to evaluate, and researched which generation people were interested in. We found that the images that have high aesthetic evaluation and originality are created in early and middle generations.

1. Introduction

The evolutionary algorithm has been employed for optimum solution searching and machine learning in the field of Evolutionary Computing, and it has achieved great results. However, there are still difficulties in applying these techniques to the art field because it is pretty difficult to conceptualize creative artifacts by solving a problem using genetic algorithm or optimization with multi-agent method.

The study in this field started with the Biomorph created by R.Dawkins (Dawkins, 1986), and there are two main types of approaches. One is Interactive Evolutionary Computing (IEC) and the other is Evolutionary Computing (EC). The former is a technique for optimization based on the user's subjective evaluation, in simple terms, the technique replaces the fitness function by human beings.

IEC method is heavily used for evaluating paintings because it is difficult to numerically express the optimization performance as an evaluation function. However it fatigues users during evolution and has many problems. For example, it cannot have many generation changes, the speed of evolution is slow, and it can't create many descendants due to the limited window space, and so on.

For these reasons, we employed EC method in this study and developed a method that simulated the real drawing process. In the drawing process, the composition and theme is decided first, and the layout for the motifs follows the theme. By employing EC, a user can set an initial image before the start of evolution, and then he can evolve it into the image he wants to output. This method has achieved rapid generation change, unlike IEC, because it only requires users to choose

an initial image without fatigue. We conducted a questionnaire for evaluating the output images using Semantic Differential, SD method to detect which generation's images are more creative. The questionnaire results show that high aesthetic and novel images were created in early and middle generations.

2. Back Ground

2.1 Previous research using IEC method

The Biomorph in *The Blind Watchmaker* by R.Dawkins is the first paper on the application to paintings with Evolutionary Computing method. It is easy to create various biological bodies like insects by repeating subjective preferences and mutations using L-system (Lindenmayer, 1968) which mathematically expresses the process of self recursive development of plants and applying the number of branches and angle parameters as genetic information. (Fig. 1)

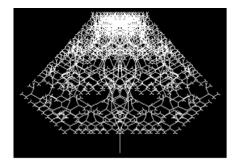


Fig. 1 Example of output images of the Biomorph

From then on, a wide variety of studies have been conducted to output paintings using many IEC methods such as morphogenesis of insects, genesis of plants based on L-system and creation of 2D graphic arts based on mathematical formulas and cellular automata. Sims (Sims, 1991) who is known as a pioneer of IEC graphic artists has evolved mathematical formulas and created images the user wants by using GP techniques and evaluating CG images corresponding to the mathematical formula. (Fig. 2). And Secretan has created genetic arts using a kind of neural networks, Composition Pattern-Producing Networks that is n-

dimensional function in n-dimensional space and using IEC method called the Picbreeder (Secretan, 2011).

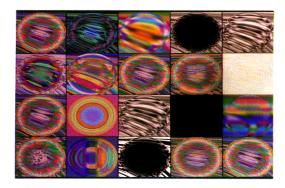


Fig. 2 Example of output images by Sims

2.2 Previous research using EC method

Most of studies on the creation of computer graphics using adaptive evolution have employed the foregoing IEC method, and there are not many studies that have used the EC method or the natural selection based on inter-component and environmental interactions. Among these studies, we focused on McCormack (McCormack, et al. 2012) who employed the process of niche formulation, and showed that each agent drew various paintings under regulated conditions (Fig. 3).



Fig. 3 Example of output images by J. McCormack

Unemi (Unemi, 1999) has created a system that locates some self-propagating individuals in two-dimensional Euclidean space and prevents them from colliding with each other during growth, and generates complex color patterns by coordinating a separate color for each individual. In addition to this, Bird et al. (Bird et al. 2008) attempted to create a robot that could draw lines as a drawing robot system. It doesn't limit the robot's drawing, but expresses "Ecological model" and is made up of the interrelationship between "Information from the surrounding environments" and "The way to reflect it to the robot's drawing". Also, Nakayama et al. (Nakayama, et al. 2012) have developed a new method that automatically creates the image in the user's mind by making cut-and-paste processing of an existing picture which has textures and colors that the user imagines into a drawing operator (Fig. 4).

2.3. Problems of IEC and EC methods

At present, IEC method is usually employed for the painting application by the evolutionary computing because it is very

difficult to conceptualize a creative artifact by the evolutionary computing and individual users have different tastes in paintings.

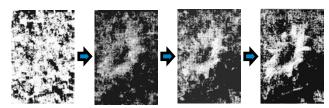


Fig. 4 Example of the evolution of paintings

The users will be able to perform the fitness calculation and the output of various paintings preferred by the individual users will be expected. However, IEC method is faced with some problems at present. First, generation change takes so much time because the user has to decide the evolutionary direction by generations. Second, it is difficult to evolve with many individuals because the pictures should be displayed on the screen by generations. Third, the user requires a certain level of training for selecting pictures, and it is especiallypronounces in abstract images that are difficult to be scored. Takagi (Takagi, 2001) has developed a technique for limiting the search range of evolution as much as possible and reducing the user's repetitive selection in order to deal with such problems. Also Auerbach (Auerback, 2012) has invented a method of automatic evolution by entering an image output from the Picbreeder into Google image search, and giving the high fitness to the frequently searched images.

However, there are still problems in determining the fitness for painting evolution on the study with EC method. As in the method above, for example, the most searched image is far from the picture that satisfies viewers. Also studies using both methods ignore the theme determination for painting. Usually a theme is firstly decided when a picture is painted by hand with painting tools. The creator starts to decide what information he wants to convey. Then, the layout of compositions and events, and coloring will be arranged in accordance with the theme. From this point of view, we consider that the IEC method is unfit for deciding a theme because the user needs to selected new image with respect to each generation, so the EC method which the user can firstly decide the initial image is fit to model the actual drawing process.

In this study, we consider "the theme of paintings" that has hardly been discussed and develop the model closely similar to the actual drawing process. In addition, we create images that provide the user inspiration for making a picture, and verify that the evolutionary computing technique is applicable to creative artifacts. Also we carried out a questionnaire survey to evaluate images output from the model.

3. Model

3.1. Model outline

We construct a system that automatically outputs paintings by use of genetic algorithm. First we get the user to select some favorite images by feeling and to decide one image as a theme. Individual piece is composed of cut-and-paste pictures and collaged images, and the fitness is given to each piece according to how much the piece is similar to the theme image. The individual piece, which is given the high fitness, can leave more images of itself to the next generation, and the piece with the low fitness is more likely to be eliminated by natural selection. Also some images randomly change by mutation.

There are three reasons that we employed genetic algorithm. First, it is easy to evolve images for many generations and it evolves them more rapidly than the IEC method. Second, the number of individual pieces in each generation is not limited, so it is possible to evolve images with a lot of individuals. Third, the user can initially decide the target image as the theme, so he can expect the exact evolution toward the theme in his mind compared to the IEC method whose images keep on evolving impromptu. Usually, the theme is decided at the beginning of drawing a picture, and then the layout of events, composition determination, coloring, and balancing in accordance with the theme are conducted. We thought such a structure that moves towards one theme or optimum solution is easy to simulate by genetic algorithm.

3.2.1 Method for vertical and lateral partitioning

We describe the method of vertical and lateral partitioning. A filial image is created by combining one half each of selected parents' images. If the parents' images are cut in half lengthwise, combine a half of each image in a horizontal direction, or if they are cut in half crosswise, combine a half of each image in a vertical direction and random numbers decides the cutting position (Fig. 5).

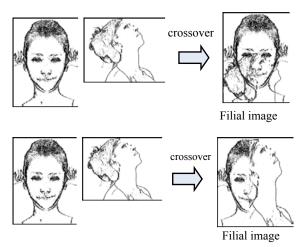


Fig. 5 (above) Example of creating a filial image by lateral partitioning method (below) Example of creating a filial image by vertical partitioning method,

3.2.2 Collage method

We describe the collage method. It means to paste the selected mother's image that is enlarged or reduced, translated, and rotated in the selected father's image. (Fig. 6 above) A random number determines how much the mother's image is deformed and where it is pasted. Affine transformation is used for changing images, and it is commonly used to deform

images because it facilitate image processing by representing image scaling, translation, and rotation.

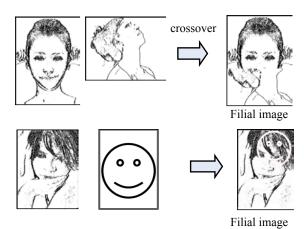


Fig. 6 (above) Example of creating a filial image by collage method; (below) Example of creating a filial image by etching method

In this study, we set the maximum magnification as 1.5 to prevent the original image from getting out of shape because of being enlarged too much.

3.2.3. Etching method

We describe the etching method. Paste the selected mother's image after the affine transformation in the selected father's image. If a pixel of the mother's image is black and also the father's one is black, change the father's one to white. In case that the mother's one is black and the father's one is white, leave the father's one as white. After this process is applied to all the pixels, the father's image can appear from the mother's image as a white picture. (Fig 6. below)

3.3. Mutation

After crossing over, some filial images mutate by the random numbers. Determine each pixel on the filial image by the threshold. If a pixel is below the threshold, it is reversed from black to white or from white to black (Fig. 7). The user can set the threshold at the beginning of the model. If the threshold is set low, the mutation is likely to occur more easily.

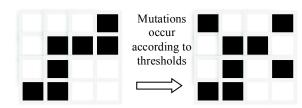


Fig. 7 Pattern diagram of image changes by mutation; left, before mutations, right after mutated.

3.4. Elite preservation strategy

The population of the next generation is stochastically formed in the selection of parent images based on the fitness of each individual. For this reason, even though good individuals are created by crossing over or mutation, sometimes they can't survive to the next generation. To avoid this consequence, a method was proposed to leave individuals that have high fitness in the population of individuals to the next generation. We refer to this as Elite Preservation Strategy.

In this model, the user can arbitrarily decide whether or not to conserve elites in the evolution. Also the user can initially decide the number of the elites conserved and the total number of individuals in each generation. The more the number of individuals, the more the probability of diverse evolutions, but it takes longer to compute, so it requires attention.

3.5. Method for evaluating the fitness

We describe the method for evaluating the fitness. A created filial image is given points according to the degree of coincidence with a target image. The target image is classified into the following three areas in accordance with the brightness value.

Area 1: If the filial image coincides with the target image, it is given D_1 point. (If it doesn't, D_2 points are deducted.); Area 2: If the filial image coincides with the target image, D_3 points are deducted. (If it doesn't, it is given D_4 points.); Area 3: There is no change whether or not the filial image coincides with the target image.

However, the Area 3 can get points if the user arbitrarily input points. When the total number of matched pixels in Area 1 is n_1, mismatched pixels is n_2, the total number of matched pixels in Area 2 is n_3, mismatched pixels is n_4, and then the fitness F of the filial image is F=D_1 n_1+D_2 n_2+D_3 n_3+D_4 n_4. The points that are given when it matches or mismatches are arbitrarily input at the beginning of the simulation.

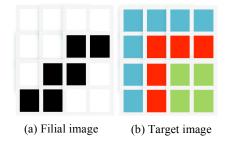


Fig. 8 Example of computing the fitness

We describe a specific example with Fig. 8 that shows a pattern diagram of a partial enlargement of an actual image. There are 4 match pixels and 1 mismatch pixel in red area, 1 match pixel and 6 mismatch pixels in blue area, and 1 match pixel and 3 mismatch pixels in green area between the filial image (a) and the target image (b). Therefore, the fitness F of the filial image is expressed as $F=2\cdot4+(-1)\cdot1+(-2)\cdot1+1\cdot6$, and then we obtain F=11.

4. Method for evaluating output pictures

4.1. Questionnaire using Semantic Differential method We classified output pictures by generations such as early, middle, and late period, and then we did a questionnaire using Semantic Differential, SD method (Osgood, 1957) for researching what evaluation of output pictures were given in

which generation. SD method is one of methods for evaluating sensibility aspects and it is used in many fields.

The feature of SD method is to evaluate a certain stimulating concept with a pair of adjectives. First, we draw up a questionnaire with pairs of adjectives that describe contrasting words and set a scale of 1 to 5 or 7 among a pair of adjectives. We evaluate a concept with respect to each pair of adjectives, and then we can compare the differences among stimuli. This method has been used for psychological experiment since early times, and deserves a long-established method. We used the studies by Tsutsui et al. (Tsutui, et al. 2001) and Inoue et al. (Inoue et al. 1985) as a reference for pairs of adjectives.

4.2. Factor analysis method

Factor analysis is a method for specifying some elements that affect a number of observed data. This element is called a factor. In this study, we regarded the questionnaire items using SD method as observed data, and used this method for searching factors that affected those results. We conducted the factor analysis using Excel 2010 (Social Research Information Co., Ltd.).

4.2.1. Correlation coefficient

We can examine the strength of relationship between two items with correlation coefficient. If the correlation coefficient is positive, the item has positive correlation, and if it is negative, the item has negative correlation. The survey of the correlation coefficient of each item suggests how the items relate to each other. It is called a correlation matrix whose correlation coefficient is calculated from each item and the coefficient is described in a matrix.

4.2.2. Identification of the number of factors

We conduct the factor analysis with the correlation matrix and determine the number of factors. The following standards are generally used for determining the number of factors.

- Gutman standard: Employ a factor whose eigenvalue is more than 1.
- 2. Scree standard: Plot the size of the eigenvalue and observe the change, and then extract the factor.
- 3. Employ the factors those of cumulative contribution ratios are more than approximately 60 %.
- 4. Employ a factorial structure with possible meanings.

In this study, we decided the number of factors as 2 factors that had high contribution ratios to interpret the results easily.

4.2.3 Factor nomenclature

We conduct a factor analysis with the number of factors determined in the previous section. We denominate the factor by reference to the factor loading of the questionnaire item that is affected by the factor.

4.2. Multiple comparison test

Multiple comparison test is used for comparing each data group without raising the significance level when there are more than 3 data groups. In this study, we compared each item in early, middle, and later generation and examined the significant difference with Tukey-Kramer method.

5. Results

5.1. Output results

We explain the output results of the model with or without elite preservation.

5.1.1. Output results with elite preservation

The following is the output results with elite preservation. We used 4 pieces of initial images (Fig5.1. (a)) and one target image (Fig5.1. (b)).

The number of individuals in each generation n=80, the mutation rate mRate = 0.000005, image size is 600×800 pixels. The score when the black area of the initial image matches that of the target image, $D_1=10$; the score when the black area of the initial image mismatches that of the target image, $D_2=-5$; the score when the blue area of the initial image matches that of the target image, $D_3=-10$ and the score when the blue area of the initial image mismatched that of the target image, $D_4=-5$. (Fig9. (c)) shows that the output results of individuals that had the highest fitness in each generation and (Fig9. (d)) shows that the fitness in each generation.





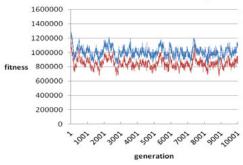




(a) Initial images

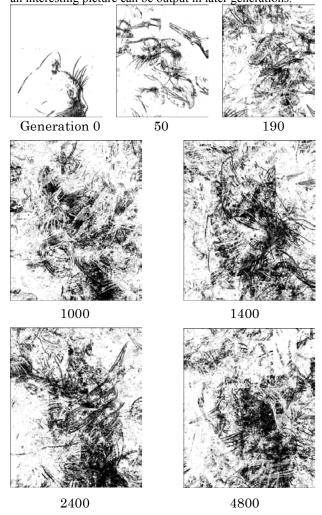


(b) Target image



(c) Graph of the fitness of each generation; red line illustrates the average fitness of the whole population, blue, the elite population.

If an individual with the high fitness is generated by crossingover or mutation, it is not passed to the next generation because the elite preservation isn't conducted. So we can see that the individual repeats evolution and selection. The feature of this case is that the initial image can't get closer to the target image but a dynamic image tends to be output. Comparing to the case with elite preservation, the picture evolution isn't fixed even though the generations roll by and an interesting picture can be output in later generations.



(d) Time development of images

Fig. 9 Output results without elite preservation: (a) Initial images, (b) Target image, (c) Time evolution of the fitness of each generation, (d) Time development of images from the initial to 4800 generations

5.1.2. Output results without elite preservation

The following is the output results without elite preservation. We used initial images (Fig. 9 (a)) and one target image (Fig. 9 (b)).

The number of individuals in each generation, mutation rate, image size, and the score when the initial image matches or mismatches that of the target image are equivalent in the preceding section. The number of preserved elite individuals in each generation is 5. (Fig. 10 (a)) shows the output results of individuals, and (Fig. 10 (b)) shows the fitness in each generation.

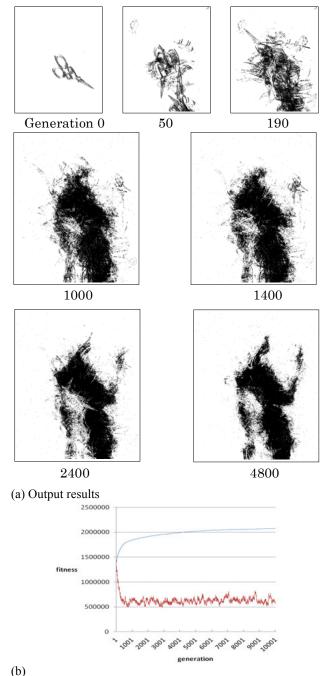


Fig. 10 Output results with elite preservation: (a) Time evolution of images, (b) Graph of the fitness of each generation; red line illustrates the average fitness of the whole population, blue line, the elite population.

If an individual with the high fitness is generated by crossingover or mutation, it is passed to the next generation because the elite preservation is conducted. Therefore, we can see the output image is getting closer to the target image as the generations roll by. Comparing to the results without elite preservation, there is less image noise, and static images tend to be output. There are defects that the picture evolution gets stacked up as the generations roll by and only the same images are output.

5.2. The results of the questionnaire

The following is the results of the questionnaire. The questionnaire was conducted at the booth of the creator's market vol.2914 in Port Messe Nagoya from 7th to 8th December 2013. The creator's market is a big event that attracts more than approximately 3000 people including professionals and amateurs for selling all sorts of original works such as fashions, interiors, crafts, visual designs, etc. We joined this event because we expected to get answers in a wide age group and have an opportunity to show a number of people output images. We got answers from 19 male and female participants between teenagers and those in their 50's in this event. The results of the questionnaire are shown as points of SD method. We prepared three sets of generated images in early, middle, and later stage (Fig. 11). Each image was printed on a paper, and they were arranged in random order.

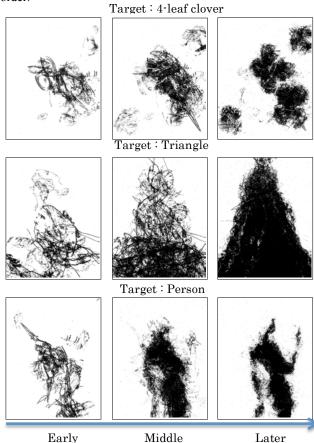


Fig. 11 Images used for the questionnaire

In the questionnaire results, high values are observed in early and middle generation, especially dynamic motion in middle generation shows extremely high values (Fig. 12). The three items "pleasure", "fun", and "originality" in early generation exceed those in middle generation, but the item "beauty" is at the same level. All the items in later generation are below those in early and middle generation, and especially dynamic motion is low.

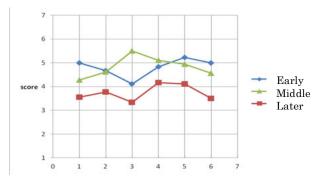


Fig. 12 Average values of the questionnaire results

5.2.1. Factor analysis

We describe the results of the factor analysis of the questionnaire. We got the cumulative contribution ratio and eigenvalue plot from the factor analysis without deciding the number of factors. The number of factor fields is numbered in descending order of the contribution ratio for descriptive purpose. If the number of factors is determined using the Gutmann standard and Scree standard, the number of factors becomes excessive, so it is difficult to grasp the structures. Therefore, we employ top two factors that their eigenvalues are more than 3 and have extremely high contribution ratios in order to facilitate the interpretation. We conducted the factor analysis of these two factors using the maximum-likelihood method and pro matrix method. We found that the first factor had high factor loadings related to aesthetic evaluation of the output picture such as pleasure and beauty in early and middle generation, and then we named this factor as "aesthetic evaluation" factor.

	Factor 1	Factor 2
(Early)		
Comfortable	0.5965	0.3899
Beauty	0.5411	0.4548
Dynamics	0.1918	0.4923
Pleasure	0.382	0.5761
Fun	0.1238	0.7694
Originality	0.0904	0.6835
(Middle)		
Comfortable	0.7263	-0.0682
Beauty	0.7626	-0.1485
Dynamics	-0.163	0.4064
Pleasure	0.6504	-0.2494
Fun	0.8858	0.0162
Originality	0.7779	0.0534
(Later)		
Comfortable	-0.2132	0.1646
Beauty	-0.5756	0.5579
Dynamics	-0.2353	-0.539
Pleasure	-0.4763	0.3311
Fun	-0.4645	0.4182
Originality	-0.1099	0.3138

Table 1. Results of factor analysis

The second factor had extremely high factor loadings related to fun and originality in early generation, the second highest factor loading was dynamics in early, middle, and later generation, and fun in later generation except for pleasure and beauty. These are related to originality and movement of pictures, so we named this factor as "originality" factor (Table 1.).

5.2.2. Multiple range test

We conducted multiple range test on each item in the results of the questionnaire and examined the significant difference (Table 2.). The results show 1% significant difference between early and later generation on pleasure items, between early and middle, and between middle and later generation on dynamic items, and between early and later generation on originality items. Similarly, there were 5% significant differences between early and later generation on beauty items, between middle and later generation on favorability items, between early and middle generation on fun items, and between middle and later generation on originality items.

(e); Early, (m); Middle, (I); Later		P value	
(e) Comfortable	(m) Comfortable	0.2733	
(e) Comfortable	(I) Comfortable	0.0014	**
(m) Comfortable	(I) Comfortable	0.0885	
(e) Beauty	(m) Beauty	0.9697	
(e) Beauty	(I) Beauty	0.0316	*
(m) Beauty	(I) Beauty	0.0556	
(e) Dynamics	(m) Dynamics	0.0056	**
(e) Dynamics	(I) Dynamics	0.3673	
(m) Dynamics	(I) Dynamics	0.0001	**
(e) Pleasure	(m) Pleasure	0.8577	
(e) Pleasure	(I) Pleasure	0.1284	
(m) Pleasure	(I) Pleasure	0.0394	*
(e) Fun	(m) Fun	0.8228	
(e) Fun	(I) Fun	0.0085	**
(m) Fun	(I) Fun	0.0401	*
(e) Originality	(m) Originality	0.6044	
(e) Originality	(I) Originality	0.0025	**
(m) Originality	(I) Originality	0.0347	*

Table 2. Results of the multiple range test

6. Discussion

6.1. Review of the results

The results of the questionnaire with elite preservation show that they came out of the opposite of the model fitness. From the table of actual factor analysis, it is believed that the fitness evenly becomes higher at every generation, on the other hand both originality and aesthetic evaluation are high in early generation, and then the originality comes down but the aesthetic evaluation goes up at the generations roll by.

And the aesthetic evaluation is decreased in accordance with the late generation (Fig. 13).

These results show that interesting images are output in early generation whose fitness is low or in middle generation whose fitness starts to increase. Therefore the model that simply evolves and aims at the target image is inappropriate. This indicates that the model that sustains the fitness in early and middle generations can output better images. The models that stochastically conduct elite preservation or change points and constrain its evolution at the time of coincidence of images when the fitness becomes excess during evolution are good examples. The results show that the case without elite preservation keeps the fitness low, but a wider variety of output images are created than those in the case with elite preservation. When Fig. 14 is taken as an example, it evolves without elite preservation using a 4 leaf clover picture as a target image and 4 pieces of initial images that are used in the section 5, the output image becomes like a witch who looks to

the distance, so there is a possibility to output a heuristic image from the target image beyond all imagination.

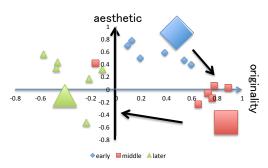


Fig. 13 Schematic diagram of the changes of evaluation by generations

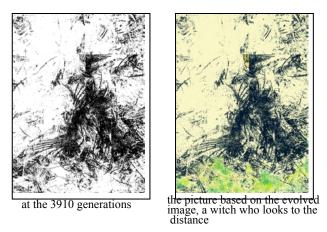


Fig 14. Pictures based on an evolved image

These results suggest that when the fitness in this model is employed, the area where we can expect an image having high aesthetic evaluation, originality, and heuristic output (Fig. 15).

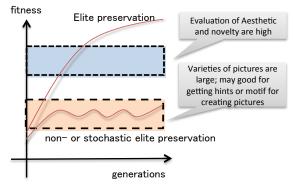


Fig. 15 Schematic diagram of each area

These are rough considerations, but we could show that extremely simple models in this study can output various pictures. These results suggest that it is worthwhile creating a model that passes through the above area and carrying out further research.

Conclusion

We have developed Natural Computing researches, and it is our challenge to understand nature as algorithm; how computing halts is a basic problem in computing natire. We believe where "aesthetics" plays a key role in computing in most of nature, halt of computing is not defined in advance, but when computing processes reach comfortable state, the process halts. What is comfort and beauty in Aesthetics has been considered, and it corresponds to "computational aesthetics" in Natural Computing context, but computational aesthetics is different from traditional Aesthetics or philosophy, comfort and beauty are considered from a viewpoint of algorithm. We have been interested in verbal feature and computing of tactile sense, and developed computing aesthetics on tactile sense proposing tactile scores for describing tactile sense as well as the pictures and their impressions in this study.

Acknowledgements

JSPS KAKENHI Grant Number 90217513 and 50177044 supported this work.

References

Richard Dawkins, (1986). The Blind Watchmaker. Norton & Company, Inc.

Aristed Lindenmayer, (1968). Mathematical models for cellular interaction in development, J. Theoretical Biology, 18: 280-315.

Karl Sims, (1991). Artificial Evolution for Computer Graphics, Computer Graphics, 25 (4) 319-328, In ACM SIGGRPH'91 Conference Proceedings.

Jimmy Secretan, Nicholas Beato, David B. D'Ambrosio, Adelein Rodriguez, Adam Campbell, Jeremiah T. Folsom-Kovarik, Kenneth O. Stanley (2011), Picbreeder: A Case Study in Collaborative Evolutionary Exploration of Design Space, Evolutional Computation, 19 (3), pages 373-403.

J. McCormack (2010), Enhancing Creativity with Niche Construction, in Harold Fellermann, et al. (eds), Artificial Life XII, pages 525-532. MIT Press.

Tatsuo Unemi (1999), A Simple Evolvable Development System in Euclidean Space, Lectures on Mathematics in the Life Science, American Mathematical Society, 26, pages 103-110, Springer Verlag

Jon Bird, Phil Husbands, Martin Perris, Bill Bigge, Paul Brown, (2008) Implicit Fitness Functions for Evolving a Drawing Robot, Applications of Evolutionary Computing, Lecture Notes in Computer Science Volume 4974, pages 473-478, Springer Verlag.

Keita Nakayama, Shinichi Shirakawa, Noriko Yata, Tomoharu Nagao, (2012) Evolutionary Creation of Painting Images using Existing Art Images, Transaction of the Japanese Society for Evolutionary Computation, 3 (2), pages 12-21.

Hideyuki Takagi, (2001) Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation. In *Proceedings of the IEEE*, volume 89, pages 1275–1296.

Joshua E. Auerbach, (2012), Automated Evolution of Interesting Images ALife XIII, (extended abstract).

C. E. Osgood, G.H. Suci and P.H. Tannenbaum, (1957), The measurement of meaning, University of Illinois presses.

Ako Tutsui, Takaaki, Shinbori and Gentarow Ohmi, (2001) Novelty and Hedonic Tone in Evaluation of Painting, *Kiso Zoukei*, 18, pages 1-6., Japan Society of Basic Design and Art.

Masaaki Inoue and Toshinobu Kobayashi, (1985) The research domain and scale construction of adjective-pairs in a semantic differential method in Japan, Japan Journal of Psychology, 33, pages 253-260.

My life as a sim: evolving unique and engaging life stories using virtual worlds

Rubén H. García-Ortega¹, Pablo García-Sánchez², Antonio M. Mora and J.J. Merelo²

¹Fundación I+D del Software Libre, Granada, Spain ²Dept. of Computer Architecture and Technology, University of Granada, Spain raiben@gmail.com,pgarcia@atc.ugr.es,amorag@geneura.ugr.es,jmerelo@geneura.ugr.es

Abstract

Stories are not only painfully weaved by crafty writers in the solitude of their studios; they also have to be produced massively for non-player characters in the video game industry or tailored to particular tastes in personalized stories. However, the creation of fictional stories is a very complex task that usually implies a creative process where the author has to combine characters, conflicts and backstories to create an engaging narrative. This work describes a general methodology to generate cohesive and coherent backstories where desired archetypes (universally accepted literary symbols) can emerge in complex stochastic systems. This methodology supports the modeling and parametrization of the agents, the environment where they will live and the desired literary setting. The use of a Genetic Algorithm (GA) is proposed to establish the parameter configuration that will lead to backstories that best fit the setting. Information extracted from a simulation can then be used to create the literary work. To demonstrate the adequacy of the methodology, we perform an implementation using a specific multi-agent system and evaluate the results, testing with three different literary settings.

Introduction

In video games, Non-Player Characters (NPCs) are a type of characters that live in the game world to provide a more immersive experience and, in some cases, present a challenge to the human player. Modern Role Playing Games (RPGs), such as The WitcherTM or SkyrimTM include hundreds of NPCs. The effort to create a good interactive fiction script is directly proportional to the number of these characters. Thus, this kind of agents usually present limited behaviors, such as wandering in the villages, selling groceries or guarding the cities. Moreover, they usually offer scripted conversations (for example, to buy and sell objects to the player), or scripted behaviors, in which they interact with the player only if he/she conducts an specific action: for example, if the player steals something then a city guard would attack him. These characters are frequently programmed to interact with the human player, following a guided sequence of activities just with this purpose, so normally they do not interact among themselves. In a world with such a number of characters, their collective interactions could improve the gaming experience, leading to a richer and more immersive world. For example, hungry inhabitants could become thieves, guards could pursuit the thieves, villagers could fell in love with others, or different war alliances could emerge.

These facts have motivated us to create a first step to achieve this objective, developing a methodology to model the language, agents and literary setting to generate backstories. In this methodology, a set of probabilities and states are associated to agents' actions, and these probabilities are optimized by means of a Genetic Algorithm (Goldberg, 1989) to match with a specific literary archetype, defined by the fiction creator. The *archetypes* are behaviors and patterns universally accepted and present in the collective imaginary (Garry and El-Shamy, 2005). They allow to empathize with the characters and aid to immerse the spectator in the story (for example, the well-known *hero* archetype).

To validate our methodology we also present a multiagent system called MADE (Massive Artificial Drama Engine) to model a self-organized virtual world where every element influences each other, following cause-effect behaviors in a coherent manner. This system must be a suitable environment for the plot of a specific literary work, also being interesting for the player/spectator.

In this work we investigate whether it is possible to model a virtual environment inhabited by hundred of characters with interesting auto-generated behaviors based on literary archetypes. Also, we test if the personality of the agents can be parametrized to obtain these different emergent behaviors. A GA will be used to find these adequate parameter values that allow the creation of quality sub-plots.

The rest of the work is structured as follows: after the state of the art, the proposed methodology is presented. Then, a complete example of application of the methodology is explained, including the experimental results for three different literary settings. Finally, conclusions and future work are discussed.

State of the art

Auto-generated interactive fiction research is mainly focused in methods to create the process of a story generation (Nairat et al., 2011). Story generation, or storytelling, can be divided in two areas: interactive and non-interactive. In the first one, and according to (Arinbjarnar et al., 2009), an interactive drama is defined in a virtual world where the user is free to interact with the NPCs and objects in an interesting experience (from the dramatic point of view), which should be different in each execution, and adapted to the user's interaction. The generation of interactive dramas can be based on a script structure (Young et al., 2004) and dramatic structures, where an inciting accident provides the motive of the drama, is followed by an increase of complications to a climax point and, finally, descending to a closure. Two examples of this approach are The Oz Project (Sloane, 2000), that requires generated narratives to follow a dramatic arc, and Façade (Mateas and Stern, 2003), that uses a structure which they call neo-Aristotelian, an adaptation of the Aristotelian structure to interactivity.

On the other hand, in *non-interactive plot generation systems* the user does not take control as the protagonist but can participate in the final result. For example, in the system presented by Pizzi et al. (2007) the user can change the emotions of the other characters but as an spectator not as an actor.

Those techniques and systems presented at the moment are focused in generating stories (interactive and non-interactive), but the present methodology, as opposed to this concept, is focused in *generating a setting* because its aim is the massive background generation for secondary characters, in order to provide a context for the writer and the player to perceive a virtual world as coherent, detailed and enriched.

The narrative is addressed by our methodology as the final step, giving freedom to creators. This issue has been studied in the systems presented in the survey by Arinbjarnar et al. (2009). Works in their survey define the plot as as something that emerges from the behavior of the agents that follow a set of rules. In the proposed methodology, the agents' behavior is produced by their personality and the environment. That is, the agents do not follow a plot, but they generate the plot itself. Furthermore, the works of the survey generate plots in worlds with a limited number of characters. This restriction does not exist in our methodology, where the number of characters to create is unlimited: they are created massively and their goal is to relate in a complex system and generate backgrounds that fit the setting of a plot, not the plot itself.

The present methodology follows the ideas of the work by Epstein and Axtell (1996), the first widely known multiagent generative social model. As a step of the methodology, a self-organizing system if defined here following the methodology introduced by Gershenson (2005): a virtual world, agents who are born, grow, interact, reproduce and dead; resources (food), mediators, and relations of rivalry (friction) and cooperation (synergy). In other step of our methodology, the actions of the agents are parametrized ac-

cording the work of Nairat et al. (2011), based in the use of GAs in order to obtain a plot (solution) where two characters interact in a creative way.

The proposed methodology is innovative since the goal has not been addressed before. Previous researches are focused on the plot, the interaction, and the narrative, but our proposal is focused in backgrounds (not in plot), where different archetypes emerge involving a starting point for future plots and subplots.

According to the taxonomy described by Togelius et al. (2011), the present methodology can be classified as a procedural content generator (PGC) mainly related to optional content, with stochastic generation and modeled as a generate-and-test algorithm (search based), that performs the optimizations of the process during the game development (offline).

Methodology

The mood, or the atmosphere, of the literary setting is one element in the narrative structure of a piece of literature. It is established in order to affect the reader emotionally and psychologically and provide a feeling for the narrative.

Our methodology defines different steps that will lead the user from the initial question "How could I obtain secondary characters for my setting, with coherent backstories consistent with its mood?" to a system that can automatically generate a setting massively populated with characters and backstories, where different desired behaviors or archetypes emerge from their interactions.

In our approach, the mood of the setting will be modeled as a group of abstract archetypes and different conditions over them. These archetypes, conceptually modeled, would be designed and instantiated as regular expressions over a language used to describe the backstories along with the social relationships between them. Those instantiations will be used by the GA to obtain the fitness of each solution. The process is iterative and presented as a waterfall model, where each step influences the following one.

The methodology includes the following steps, as shown in Figure 1: Modeling (the language, the literary setting, the agent), Definition of the GA characteristics, Instantiation, Execution and Interpretation, which will be exposed in the following subsections.

Modeling

The Modeling step affects different elements that make up a system where different agents generate words, lately interpreted as backstories, whose symbols belong to a language that also defines the desired archetypes.

Modeling of the language Each agent has to be modeled as a Finite State Machine (FSM) (Booth, 1967) whose transitions generate symbols in a language based on the following one, expressed in Backus-Naur Formalism (BNF):

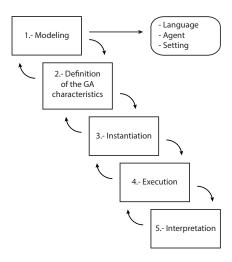


Figure 1: The proposed methodology for designing literary backstories

Starting from this partial grammar, the user has to define the expressions:

- <date>: Should be expressed in a way that is meaningful to the type of background that we are looking for. For example, if we are talking about persons, it could be the age of the agent expressed in days.
- <action_id>: Different actions can be taken into account, and will depend on the kind of archetypes we will look for. For example, if we are trying to find archetypes about love like the classical Romeo and Juliet one, where two characters are in love but belongs to families in war and finally ends up with their death, we could include actions and relations like "loves", "hates", "is son of", "is daughter of", "is parent of", "die", etc.
- < direct_object>: The possible values of the direct objects depend on the nature of the actions.
- <indirect_object>: Should define the unique id of an agent or a group of agents.

The backstory generated by each agent using the language modeled this way will be automatic and human readable. Eventually, the backstories will evoke archetypes and behaviors. As we will see in the next sections, we will try to promote the emergence of specific archetypes in order to retrieve the best backstories for our setting.

Agent modeling In a top-down approach (the opposite to the one proposed in the current methodology), if the creative needs a second character or an extra for a story, he or

she creates it out of the blue, with the background that better fits with his/her needs. This approach, valid for a small amount of characters and poor requisites of coherence between them, becomes more and more complex when we add more characters to the environment, because each addition implies new relations in their social network.

Otherwise, the bottom-up approach promotes the agent as one of the pillars of the system because it offers the coherence of the backstory regarding to the timeline (for example, an agent cannot die before he or she is born). Some actions need to be sequential and to the relations (many actions involve two different agents). An agent is just a simple tool to generate a background, a character. In an environment where many characters can "live", make decisions taking into account the other characters and affecting them, the generated background becomes complex, many links have been created and no creative process has taken part on it (the creative process is present in the definition and modeling of the agent itself).

Given this premise, a multi-agent system seems promising for generating this kind of complex relations and backstories.

An agent can be seen as a Finite State Machine (FSM), so the multi-agent system relies in the sequential executions of time portions. Every execution implies a review of the current state and (maybe) one or more actions done, depending on local and external properties.

The agent should contemplate different states depending on the type of information we are interested in ("alive", "dead", "pregnant", etc) and should also have local properties that define the possibility to make certain decisions.

Agents live in an *environment*, understood as the virtual spatio-temporal frame where the agents play their lives. This environment is also responsible for executing the main process of each agent (corresponding to a portion of time). Each iteration the agents should be chosen randomly or based in classical role-playing games features like "initiative". Also, the environment provides a set of functions to interact with other characters and the environment itself.

The key in the use of agents to create background for characters relies in the following fact: Some actions are performed statically depending on the inputs and the current state but other (the majority) also depends on agent's local features and probabilities. Even if all the agents share these values, it is difficult to predict the result when the system is so complex. A minimal change in one probability to make one decision can imply completely different scenes. For this reason, some actions should be statically defined and others have to depend on initialization properties. We define two kind of properties:

• Base properties: Those that are intrinsically defined by the nature of the conceptual agent. For example, if we

are modeling a simplification of a person, the average life expectancy could take values from 70 to 85 years. An agent that lives 100 years would be unusual, and an agent that lives 200 years should not be possible.

 Searchable properties: Those used to increment or decrement the base values in the established limits or those that are used directly as probabilities to make decisions.

Due to the complexity of the system and the introduction of probabilities, even if all the agents have the same features and probabilities, the sequence of the actions for every agent becomes unpredictable. For example, if we have our agent modeled and we choose random values for the configuration, roughly, the characters' backstories will be "normal", showing more or less the stereotyped behaviors (that make the character a group representative rather than an individual), but maybe, some of them show higher level non-modeled behaviors. In the next subsection, we will provide a way to model those high-level behavior in the way that, given a executed environment, a computer is able to find them and, in the Execution step, obtain a configuration that promote the apparition of these archetypes in the execution.

Literary setting modeling The setting of a story is defined as the historical moment in time and geographic location in which it takes place, and helps initiate the main backdrop and mood for a story. Conceptually, the setting of a story is, in this methodology, independent from the agent design, but has to agree with it in the selected language: The agent generates actions in a previously defined language with a clear semantic interpretation. On the other hand, the setting is composed by criteria over patterns for this language and the social networks derived from its usage. In other words, the agents create backstories and the settings matches specifics patterns inside these backstories.

For example, if we have a medieval storytelling setting and we have defined a language where the actions "love", "hate", "attack", "defend", "win" y "lose" are used, we could try to find classic archetypes like the "villain" and the "hero". A villain could be a character that hate many others, attacks them and win. A hero could be a character who is loved by many people, that eventually defends them and then attacks to the villain. Moreover, if the mood of the story is sentimental, we could be interested in a "heartbreak and meet again" archetype, where two characters are in love, after that they hate each other and finally fall in love again.

In our methodology, an archetype is designed as a function that receives the backstory of an agent, processes it, tries to find patterns and interpret the social network related to the agent, and returns a *true* value if the agent matches the behavior. The complexity of the agent is not evaluated, just the result of its execution in this environment. It is important to remark that the agent has to be modeled to play a normal

life (stereotype), and that the archetypes work as high level behaviors not directly implemented.

A setting can be seen as a function over the agents that matches an archetype. If the archetypes emerge in the desired way, the setting function would return high values.

In the next section, we will explain the mechanism that will optimize the agent's searchable features to obtain backstories coherent with the mood of the setting.

Features of the Genetic Algorithm

The codification of the chromosome of the individuals of the GA consists in mapping the searchable properties to an array of values.

The fitness function of a chromosome is the result of calculating the average of the application of the setting function over n executions of the environment with defined base properties (where n is the minimum number that reduces the error and has to be calculated empirically).

It is important to clarify that the term "individual" in this work refers to a solution in the GA that models a whole environment, where a number of different "agents" are living. So, an "individual" in the GA is not equal to an "agent".

At this point, we remark the possibility of using different number of *profiles*. In this context, a profile is a set of properties assigned to an agent. If two agents have different profiles, their behaviour in the face of the same inputs can be different. In the chromosome, the use of n profiles means a chromosome size equal to the number of searchable properties multiplied by n.

Intuitively, increasing the number of profiles will lead to richer backgrounds, but *a priori*, since the system is complex, it is very difficult to establish which number of profiles will lead to the fittest solution without empirical tests. In some cases, a small number of profiles can generate many different archetypes and the other way around. Moreover, it is important to remark that the bigger the chromosome is, the slowly the solution converge.

Instantiation and execution

After the Genetic Algorithm is configured, some properties need to be established in order to fit to the desired setting:

- Base properties: Including the threshold for the agent's features, and environment parameters (i.e. size of the world, resources, etc).
- Genetic Algorithm parameters: selection, genetics operators (crossover and mutation and termination condition)
- Number of profiles

Once the parameters for the GA, the environment, the agent and the number of profiles are set, the GA can be executed. In some cases, the fitness can be improved by modifying base parameters, the evaluation of the archetypes or the logic of the agent. The process is iterative and a fine

tuning is essential to obtain the best fitness. Once the best solution has been found, the environment can be executed and the backstories generated can be used.

Interpretation

Due to the nature of the grammar proposed for the actions, they can be directly converted to natural language. Applying the setting function (used as fitness by the GA) to the executed environment could be useful to tag the backstories with the different archetypes found and their explanations. As an example, if our setting is the "far west" and we need non-player characters to populate a saloon, we may want to find a "Billy the kid" archetype and three "player" archetypes. If they are met by the main character he/she may discover their backstories, that would be coherent with the world they live in and the mood of the story.

Applying the methodology

To validate our methodology we apply its steps using a specific scenario and environment. In this scenario we want to model the next story: - "A number of rats live, eat, reproduce, compete for food and death within the walls of the Invisible University of Ankh-Morpork¹. As the University professors, these rats are very vindictive and territorial."

Agents in the MADE environment

The first step is to model the agents and their environment. In this work we propose the MADE (Massive Artificial Drama Engine for non-player characters) environment, a virtual place where different agents play their artificial lives. Its functions are to initialize agents in the map, control the time, execute the agents during a time unit (for example, a day in the story) and update the map.

The MADE environment can be configured by using the following parameters (with the values that will be used in the experimental section in brackets): Number of agents initially placed (15), map square grid dimension (10), number of rations randomly placed in the grid each day (10) and duration in virtual days of the execution of the environment (1000). These parameters can affect directly to the behavior of the agents.

A MADE Agent lives in a MADE Environment, occupies a cell in the grid, moves around looking for food or mates and interacts with other agents.

The design of a MADE Agent has some restrictions:

 An array of parameters (probabilities represented as real numbers between 0 and 1) should be provided in its initialization. These parameters should be used by the agent in its day-by-day decisions.

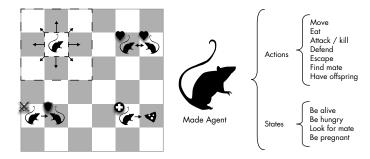


Figure 2: Actions and states modeled in the MADE Agent.

- Every action is subject to be logged for later analysis.
- As a result of some iterations of the agent's day-based life-cycle, the agent should present the behavior of a *liv-ing thing*, that is born, eats, grows, reacts, reproduces and dies.

A very simple agent has been designed for this study: a virtual rat, that models:

- Four states (be alive, be hungry, look for mate and be pregnant) that represent internal situations that will lead the agent to perform the actions described in the item bellow.
- Seven actions (move, eat, attack, defend, escape, find mate and have offspring) that lead to a basic instinctive animal behavior, very useful for this work since it can be the canvas of complex *humanized* behavior patterns.
- Parameters that define the characteristics and probabilities to perform actions depending on the state.

It is important to remark that no "feelings" and no "memory" have been modeled in the MADE agent for this study. This is illustrated in Figure 2.

Every decision made by the agent is based on its state and its characteristics (probabilities to perform different actions).

The MADE Agent is created using twelve parameters, that define its base features and probabilities to make the decisions presented in the state diagram in Figure 3. The execution of an agent is dynamic, and depends on the internal probabilities and states but also on the neighborhood, and the map configuration. Even so, we can say that these initial parameters define in some way the possible situations where the agent could be involved.

The source code of the MADE environment and the algorithms used in this work are publicly available in https://github.com/raiben/made under a LGPL license.

¹This is our tribute to writer Terry Pratchett, whose books have inspired us to create these agents.

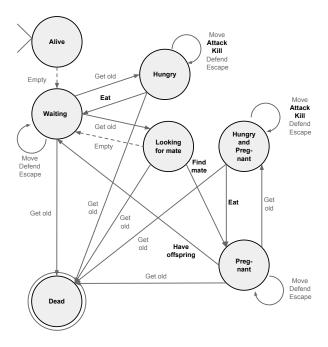


Figure 3: MADE Agent's finite state machine

Definition of the literary setting

As previously said, agents are independent of the desired literary settings. To validate our approach three different literary settings are going to be tested.

The first (and simpler) literary scene is called "revenge" and its goal is to model individuals with complex memory-based behaviour between two characters (although the agents in our environment have not memory). It tries to find the number of profiles and values which are optimal to make *revenge* archetype emerge in as many agents as possible after 1000 days. An agent (a) will be considered as an *avenger* if it has been attacked by other agent (b) and after that, in a moment in its life, it has satisfactory attacked the agent b, in *revenge*. The value of the days is set to 1000 because is a duration long enough to make the archetype emerge.

The secondary literary scene, "territorial war", aggregates different sample archetypes where many factors must be taken into account. It tries to find the number of profiles and values that generate at the end of the run an equal distribution of the archetypes downtrodden (an agent that has been attacked at least two times and has defended the position), warrior (if it has satisfactory attacked at least five times), helpless (if it has been attacked at least ten times and has not defended the position) and bad warrior (if it has been defeated at least ten times).

Also, after 1000 virtual days, the alive population will be the 60% of the total population (archetype *survival popula*- *tion*). We have used the presented values to define this scene because, in our opinion, they model an interesting literary setting.

Finally, the last literary setting, "Shakespearean", models different archetypes present in the works of Shakespeare to create a richer and complex scene. In this scene, there exist several well-known archetypes, proposed in the work of Vogler (1998). The first one is *star crossed lovers* represents the Romeo And Juliet: rats who have offspring and whose parents have participated in a fight. Also, other archetypes must arise in this setting: the *shadow* (an agent that kills another), the *hero* (an agent that defeats a *shadow*), the *mentor* (an agent that gives food to the *hero*) and the *threshold guardian* (an agent that attacks the *hero* at least two times). The last two archetypes can not be *hero* or *shadow*, adding complexity to the environment.

Definition of the genetic algorithm

As previously said, the parameters used to define the agents are mapped into a chromosome, and a Genetic Algorithm is used to evolve the solution. The fitness function (or setting function) is expressed in terms of:

- Regular expressions applied to the log of each agent in the environment: An agent is tagged when a regular expression matches its log.
- A numeric function over the number of tagged agents for each archetype: the fitness of the solution is incremented with the returning value.

Different fitness functions have been used. In the first literary setting ("revenge"), every agent whose log matches the archetype adds one point to the fitness. Therefore, the goal is to recreate an environment where several "avenger" agents exist.

To model the "territorial war" setting we have defined the fitness function as follows: if the exact percentage of agents are tagged with one archetype defined, 1 point is added to the fitness. The maximum is therefore, 5 points. However, all the fitness values use a normal distribution over the percentage of appearance. For example, with the archetype "survival population", the maximum value (1) is obtained when the 60% percent of the population is alive, and the normal distribution begins in the 30% and ends in the 90%. For the rest of the archetypes, 1 point is added if the 22'5% of the population is tagged with each one of the archetypes, and each normal distribution begins in the 8% and ends in the 30%.

Finally, the function for the "Shakespearean" scene is the sum of the proportions of each archetype with respect to all the population. Therefore the maximum would be 5 (if the archetypes were not exclusive).

Thanks to the agents' logs, we can know every event (internal and external) of their lives, and evaluate their interest or adequacy to a specific literary setting.

Parameter	Value	
Codification	12 alleles per profile	
Fitness function	Average of 10 executions.	
Natural selector	Original Rate: 0.9	
Crossover operator	Rate: 35%	
Mutation operator	Desired Rate: 12	
Stop condition	100 generations	
Population size	30	

Table 1: Parametrization of the Genetic Algorithm

In this work, we have implemented a method based in regular expressions with backreferences. The proposed technique puts annotations in every agent whose log matches a complex regular expression able to find emerging high level behaviors, not implemented in the life-cycle. In this case, we do not exactly know how many roles are necessary to create a world of "warrior", "vindictive" or "shadow" rats, so different number of profiles (from one to five) will be considered.

If only one profile is used in a run, all the agents are created with the same parameters, evolved by the Genetic Algorithm. If more profiles are used, they are assigned to the agents in order of appearance in a loop. Our assumption is that some archetypes could emerge using one profile and other will need more (those that require two clearly differentiated roles). It is important to remark that the number of alleles of the chromosome are multiplied by the number of profiles, so the convergence of the solution will be be affected by the number of profiles used.

Execution and results

For the experiments performed in this work, we have used the parameters shown in Table 1. These values have been chosen empirically after several test runs.

The results of the experiments are shown in Table 2. It shows the average of the best fitness and the average population fitness at the end of 30 executions for each configurations. Boxplots of the best fitness obtained are shown in Figure 4.

With respect to the first literary scene, the Kruskal-Wallis and Wilcoxon pairwise comparison shows significant differences among all configurations (p-value << 0.05) except between P2 and P3 (p-value=0.3). Therefore, we can conclude that in this kind of global archetype only a profile must be used for obtaining the best results. This makes sense, because we are looking for one type of local archetypes (avenger), so adding extra profiles leads to different behaviors of the agents.

The results of the second literary setting ("territorial war") shows differences among all the number of profiles (p-value <<0.05). As we suspected, it is clear that using one pro-

Profiles	Setting 1	Setting 2	Setting 3
1	495.513 ± 20.091	0.765 ± 0.037	2.584 ± 0.044
2	471.206 ± 24.550	1.063 ± 0.115	2.433 ± 0.145
3	455.42 ± 28.240	1.093 ± 0.063	2.336 ± 0.141
4	431.926 ± 31.682	1.084 ± 0.048	2.281 ± 0.052
5	411.24 ± 25.023	1.045 ± 0.110	2.266 ± 0.068

Table 2: Results for 30 executions of each configuration using 1 to 5 profiles (average best fitness \pm std. dev).

file is not enough to emerge the desired archetype. However, the pairwise comparison using Wilcoxon does not find significant differences using more than 2 profiles. This can be explained because an agent could share more than one archetype at the same time. A promising number of profiles could be 4, because their only lower outlier is not as distributed as the others.

Finally, the results of "Shakespearean" scene surprisingly show that using only one profile is the best choice to generate a complex world as the one desired, decreasing with the number of profiles (except for 4 and 5, where no significant difference exist, with p-value>0.05). This can be explained because, to model all the archetypes, all actions available in the environment should appear. Therefore, a profile optimized to trigger all available actions would generate more archetypes. Also, and as previously said, using more profiles requires more time to converge, as we are using larger chromosomes. Furthermore, the actions of our environment could be quite simple to model complex stories with different personalities.

Convert results to literary language

Every agent has a log that stores all the relevant events in its life in a simple format. Each line of the log indicates the day, the event in a short readable format and some extra information. Every agent's log in the MADE Environment is coherent with all others' logs. Many plots are being created, with a structured format that can be read by game engines or natural language processors. This log can be used for evaluation, for example, obtaining feedback from human readers.

Conclusions

This work presents a general methodology to design emergent literary stories in massive virtual worlds. The described steps include the modeling of the agents and literary setting. Then a Genetic Algorithm is used to optimize the parameters of the agent's profiles (behaviors) using as fitness a function that models the literary setting.

Given a literary setting, an author of a story or a video game could define different rates of archetypes or behavior patterns, and use the techniques described in the present work to obtain the optimal profiles. In this work, several profiles have been generated in the MADE (Massive Drama

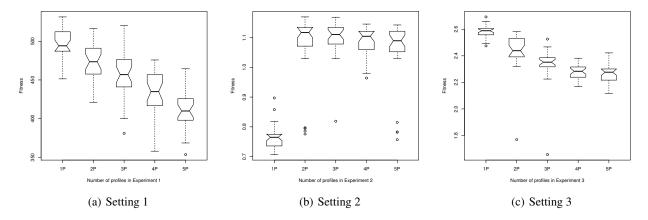


Figure 4: Average fitness of the 30 best individuals of the GA for each setting.

Engine for non-player characters) Environment to produce a background (or set of characters' lives) where the archetypes have emerged and have automatically created massive backstories coherent with the settings of the artwork.

In future works, more complex agents will be used, with different rules to be modeled. For example, we plan to model more human behaviors such as love or envy, to generate interesting plots such as wars, weddings, or family crimes. More different fitness functions will be tested, for example, taking into account human opinions to establish the interestingness of a generated plot. Also, this system will be tested into an existent and well-known game, such as SkyrimTM, whose AI engine is publicly available for players and researchers.

Acknowledgments

This work has been supported in part by FPU research grant AP2009-2942 and projects SIPESCA (under Programa Operativo FEDER de Andaluca 2007-2013), and TIN2011-28627-C04-02.

References

Arinbjarnar, M., Barber, H., and Kudenko, D. (2009). A critical review of interactive drama systems. In *AISB 2009 Symposium*. *AI & Games, Edinburgh*. Citeseer.

Booth, T. L. (1967). Sequential Machines and Automata Theory. John Wiley and Sons, Inc., New York, 1st edition.

Epstein, J. M. and Axtell, R. L. (1996). Growing Artificial Societies: Social Science from the Bottom Up (Complex Adaptive Systems). The MIT Press.

Garry, J. and El-Shamy, H. M. (2005). Archetypes and Motifs in Folklore and Literature: A Handbook. M.E. Sharpe.

Gershenson, C. (2005). A general methodology for designing selforganizing systems. *arXiv* preprint nlin/0505009.

Goldberg, D. E. (1989). *Genetic Algorithms in search, optimization and machine learning*. Addison Wesley.

Mateas, M. and Stern, A. (2003). Façade: An experiment in building a fully-realized interactive drama. In *Game Developers Conference*, *Game Design track*, volume 2, page 82.

Nairat, M., Dahlstedt, P., and Nordahl, M. G. (2011). Character evolution approach to generative storytelling. In *Evolutionary Computation (CEC)*, 2011 IEEE Congress on, pages 1258–1263. IEEE.

Pizzi, D., Charles, F., Lugrin, J.-L., and Cavazza, M. (2007). Interactive storytelling with literary feelings. In *Affective Computing and Intelligent Interaction*, pages 630–641. Springer.

Sloane, S. (2000). *Digital fictions: Storytelling in a material world*. Greenwood Publishing Group.

Togelius, J., Yannakakis, G. N., Stanley, K. O., and Browne, C. (2011). Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):172–186.

Vogler, C. (1998). The writer's journey. Wiese.

Young, R. M., Riedl, M. O., Branly, M., Jhala, A., Martin, R., and Saretto, C. (2004). An architecture for integrating planbased behavior generation with interactive game environments. *Journal of Game Development*, 1(1):51–70.

Multiagent System Architecture in Orphibs II

Nuno Barreto¹, Luís Macedo¹ and Licínio Roque¹

¹ CISUC- Centre for Informatics and Systems of the University of Coimbra

Department of Informatics Engineering

University of Coimbra

{nbarreto, macedo, lir}@dei.uc.pt

Abstract

Orphibs II is an Artificial Life game where the player takes the role of a caretaker of alien-like creatures called Orphibs. Each orphib is an autonomous agent that has its own personality, given by internal utility functions that can be genetically transmitted. In this paper, we present the architecture behind the agents present in the game we developed, as they feature a new mechanism built on top of a Goal-Based Behavior approach: genetic personalities. Genetic personalities introduce behavior variability for A-Life games and account for new gameplay mechanics such as artificial behavior selection. We start by surveying the state of the art regarding approaches to Multiagent Systems in videogames, in general, and Artificial Life games, in particular. We then describe the architecture behind the agents in Orphibs II, including their reasoning system and evolutionary mechanism. Finally, we present a discussion and potential future work.

Introduction

Inspired by biology, Artificial-Life (A-Life) games are a subgenre of Simulation games that simulate virtual life, where the main gameplay focus is on the interactions that the game's entities (usually represented as virtual living beings) have with the world as well as with the player [1]. Typically, A-Life games provide the player with a sandbox that can be used as a test bed for biological and, to some extent, social experiments. Several games, including Creatures [2] and The Sims [3], have shown that A-Life has market appeal and, at the same time, incites scientific curiosity [2]. There is also a research field behind A-Life which aims to develop techniques and approaches to create virtual agents that appear to be alive, or even sentient.

In this paper, we present a thorough analysis of the agent architecture and the mechanics behind Orphibs II, a game we developed (created to showcase the tools in [4]). Orphibs II is an A-Life prototype game where the player takes the role of a creature caretaker. It borrows some elements from Creatures and The Sims, such as creature evolution and goal-oriented behavior, but, at the same time, introduces new mechanics, namely the use of utility functions that can be genetically transmitted to account for genetic personalities. Moreover we provide a simple experiment to verify how a population of randomly generated Orphibs behave, i.e., if there any emergent behaviors and, in addition, the evolution of the population throughout the experiment.

The main contribution in this implementation lies in Genetic Personalities: an improvement upon the regular goal-oriented behavior which allows for behavior variability, as well as accounting for new gameplay mechanics such as artificial behavior selection.

This paper is structured in the following manner: Section II will present the State of the Art of A-Life games and approaches, Section III overviews Orphibs II and especially its architecture; Section IV showcases an experiment that we conducted as well as its results; and Section V describes future work; and, finally, Section VI, presents the conclusions.

State of the Art

Internally, A-Life games are composed of Multiagent Systems (MAS)which model the living beings the game is attempting to simulate. As such, agent development in this type of games follows an Agent Centric, or Bottom Up [5], approach. This is a method where the agents are entities that populate the game world. Here, agents are created with an internal reasoning system that allows them to interact with the world. Since each agent's reasoning system is independent from the others', this approach supports emergent behaviors. There is another agent development approach in videogames, the System Centric one, that is beyond the scope of this paper, because this type of approach is best suited for tactical behaviors [6] or game engine components [7].

While there are some techniques, including Finite State Machines [8], Behavior Trees [8] and Fuzzy Logic [8], which are commonly used in other genres, the A-Life genre employs techniques such as Rule-Based Systems [8], Neural Networks [8], Cellular Automata [8], BDI Architecture [9] and Goal-Oriented Behavior [8, 10].

Rule-Based Systems (RBS) [8] are a model in which the designer defines a set of condition-action pairs that result in simple behaviors. RBS are present in [11], where the authors developed a MAS to account for the emergence of circadian rhythms through evolution. Although most of the experiments showed that the agents did not present a circadian rhythm, they did, however, demonstrate interesting emergent behaviors. Aylett and Cavazza [12]illustrate another application for RBS, in their work. They explain that rule based systems have been used in crowd behavior. A sub-set of RBS, Cellular Automata [8] are a discrete model that consist in a grid of cells, where each cell has a value attributed

according to a set of rules that takes into consideration its neighbors' values. Conway's Game of Life [13] is a direct usage of this approach.

Neural Networks (NN) [8] are inspired by how the brain's cellular structure works. An artificial brain is composed of layers of artificial neurons with weight links between them. In order to calculate the links' weights, feedback mechanisms may be used at run time, making this approach suitable for adaptive behaviors. The main disadvantage of NN is that they are a black box model. This means it is not clear to a human how a given Neural Network works, thus making it hard to debug. For this reason, they are not used in most games. In Creatures [2], agents use an architecture which includes NN to allow them to learn activities such as walk and communicate with the player.

Belief-Desire Intention (BDI) Architecture [9] is another approach in modeling agents. It was created to be akin to how humans make decisions. In this architecture, an agent contains a desire, belief and intention set. Beliefs represent the agent's perceptions, desires symbolize preferable end-states and intentions are the list of possible actions. This architecture has additional functions: the belief update function that allows beliefs to be refreshed according to new perceptions, the options function that uses agent's beliefs and intentions in order to produce desires and the filter function in which beliefs, desires and intentions are used to generate intentions. These mechanisms underlying the BDI architecture result in a plan, or a set of actions, for an agent to follow. This architecture, in conjunction with PSI psychological theory [14] for the display of emotions, is used by [15] with the purpose of creating autonomous non-playable characters in an educational Role Playing Game. Other uses of the BDI architecture include the design of agents present in Unreal Tournament 2004 [16] by [17] and [18]. In both cases, the authors argue that this approach yields agents with humanplayer-like behavior.

Goal-Oriented Behavior [8, 10]can be seen as a simplification of the BDI Architecture. Here, agents have a set of goals, usually depending on internal needs, and a set of actions. In this method, the agent performs actions to fulfill its current goals which, in turn, are chosen according to the agent's current needs. This method also supports utility functions. In this case, goals are ranked according to a function that maps needs to a score, called utility. As [10] explains, this is the technique used by The Sims videogame series.

A-Life Games

Although the A-Life genre is not as popular as others, there are several accounts of commercial and non-commercial games as well as research projects available.

Introduced as a mathematical game, the Game of Life, created by John Conway [13], consists in a 2D cellular automaton with a specific rule-set that states that each cell can either be dead or alive given its internal state as well as its neighbors'. The game can be played by creating the automaton's initial state and observing the results.

In 1994, Sims [19] developed a system where creatures would compete against each other in order to evolve morphologies and animation controllers that would adapt to a given environment. This work would later serve as basis for

the game called 3D Virtual Creatures Environment [20, 21] that works in a similar manner.

Creatures [22, 2], developed by, among others, Grand is a game where the player takes care of virtual creatures called Norns. Norns can learn to adapt to their environment through NN linked to sensors and an artificial hormone system [2], Grand et al. even argue that social behaviors have been observed to emerge in Norns, such as playing collective sports yet they explain that due to the blackbox nature of NN, they could not verify the veracity of their claims. Recently, Grand has begun developing an A-Life called Grandroids [23]. Unlike Creatures, Grandroids are meant, according to Grand, to have a more complex artificial brain.

Little Computer People [24], one of the earliest Human A-Life commercial games, is a game that simulates the life of a single person inside a two-story house. The person acts autonomously and can even interact with the player through verbal communication and a poker mini-game. The player, on the other hand, can also persuade the virtual person to perform some actions through simple text commands.

Inspired by Little Computer People, The Sims [3] is a well-known A-Life series of videogames. Here, the player influences the lives of virtual people called Sims. Zubek [10] explains that sims are driven by needs. Each need, or drive, has an underlying function that converts drives into a global happiness value. Sims then select their next action in order to maximize their happiness. The Sims 2 also introduces the concept of personalities, which are implemented by attributing different weights to drives.

Another well-known series is Petz [25]. Petz is composed of the subseries Dogz, Catz, Babyz, Oddballz and others. Each subseries follows the same basic gameplay structure present in A-Life games: the player has to take care of pets or humans (both called petz by the creators), including dogs, cats, babies and alien-like creatures, while they develop. Petz are able to perform a variety of actions as well as display emotions through body language and facial expressions [26]. Internally, this is done through a set of mechanisms that control a specific feature, including breathing, emotions and multiple planning. Additionally, the Babyz subseries also features learning mechanisms [27] so babyz can learn to understand simple words.

Unlike the previous games, Chao Garden [28, 29] is not a game in itself, but rather a meta-game present in both Sonic Adventure and Sonic Adventure 2. Chao Garden introduces a different gameplay mechanic: the ability to train the game's virtual creatures, called Chao, for competition, by maximizing attributes such as strength and stamina. While independent, Chao cannot develop by themselves the attributes necessary for competing and require to be given special items by the player. Other particularity of this game is the fact that Chao can develop personalities according to the way they are treated, or mistreated.

Finally, Species: Artificial Life, Real Evolution [30] is a game where several species of creatures walk around a procedurally generated virtual world. The player can then feed them, kill them, and even extract their DNA to splice into another creature's DNA. Nevertheless, without the player's interaction, the creatures can still be part of a natural selection process.

Orphibs II

Game Overview

Orphibs II is a game inspired by Creatures and The Sims where the player takes the role of a caretaker for alien-like creatures called Orphibs. Much like other A-Life games, the player does not control an avatar but rather is represented by a hand that can pick-up objects, including Orphibs, scroll through the game world and observe the internal state of the agents. Figure 1 shows a screenshot of the prototype game:



Fig. 1. Screenshot from Orphibs II. It illustrates two orphibs (one of them showing its attributes), an egg and several of the in-game objects: toys, an apple and the straw bed. It also shows one of the Orphibs' satisfaction levels as well as its current action.

The game takes place in a 2D top-down forest-like pathway map where both the player and the orphibs are constrained to the map's limits. The game world is populated by several objects including a square straw bed, a tree that produces a random number of apples at a constant rate and three toys. Orphibs can use each of these objects except for the tree. Additionally, the world is subject to a day-night cycle.

Orphibs Mechanics Overview

Orphibs are anthropomorphic bipedal oviparous and genderless creatures. They have some internal needs [8], called drives, that influence their actions. These needs are a simplification of the drives introduced in The Sims [10]. As such, Orphibs feature one physical need, hunger, and three mental needs: energy, fun and social. In addition, Orphibs have the need to reproduce.

Drives regulate which actions the orphibs can perform at a given time (see subsection Reasoning System for additional details). Currently, Orphibs can play with toys, eat apples, sleep on the straw bed, talk to each other, wander around the game world and reproduce. Whenever an Orphib performs an action that requires it to hold an object, it first travels to the object's position then it checks if the object is not being used by any other Orphib. If the object is indeed being used, the Orphib waits for its turn or looks for something better to do. This is accomplished with the help of an eyesight system that

varies according to the orphibs' eye colors and the world's light intensity at a given time.

As previously mentioned, Orphibs are oviparous and, while they are genderless, they feature a sexual reproduction mechanism. This means that when they reproduce, they lay eggs which result in an offspring that receives genetic material from each of its progenitors (subsection Evolutionary Mechanism provides a more in-depth analysis of the Orphibs' reproductive system).

The Orphibs can also mature over time. They grow into a maximum height given by the function presented in Figure 2. This function was created empirically using data-points to simulate a growth rate that is not linear. Additionally, when growing, Orphibs display body discoloration.

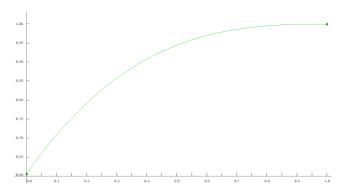


Fig. 2. Screenshot of the Growth Function in Unity3D's Curve Editor. The x-axis, ranging from 0 to 1, represents the Orphib's relative age, while the y-axis, also ranging from 0 to 1, represents the Orphib's relative height.

Orphibs also die during their lifetime. Every 5 seconds, a probability check is done for each Orphib by generating a value from a random uniform distribution and comparing it with the result of the death probability formula, which is given by a linear interpolation of the Age-Death Probability (AD) function and Energy-Death Probability (ED) function, expressed in the formula p = 0.99*AD(x) + 0.01*ED(y). It is worth noting that both the functions and the interpolation factor where chosen empirically as a game design decision in order to avoid Orphibs to die too frequently. Figure 3 and Figure 4 illustrate the curves for both the ED and AD functions respectively:

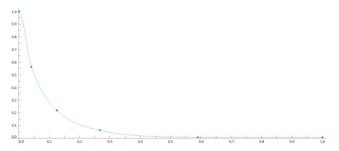


Fig. 3. Screenshot of Energy-Death Probability in Unity3D's Curve Editor. The y-axis represents the probability of dying, while the x-axis represents the energy need values.

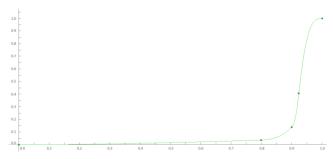


Fig. 4. Screenshot of the Age-Death Probability in Unity3D's Curve Editor. The y-axis represents the probability of dying, while the x-axis represents the Orphib's relative age.

Finally, when the player places the cursor above an Orphib, its attributes are shown on the screen. More specifically, the player can observe the creature's drive levels, in color-coded bars raging from green to red according to the drives' values, as well as the Orphibs' current and future actions.

Agent Architecture

The Orphibs II prototype was created using the Unity3D [31] game engine and, as a result, its architecture is component-based. On a side-note, due to the scope of the paper, this subsection will mainly focus on the agents behind the Orphibs.

Internally, each Orphib, game item (apple, tree, bed, egg and toys) and the map are composed of a hierarchy of Unity3D's GameObjects. For instance, the Orphibs contain a balloon object displaying the creature's current action, an object with the graphical representation of the satisfaction levels and a model object comprised of the creature's body parts for skeletal animation purposes.

Architecture Diagram.Figure 5, illustrates an Orphib's internal agent architecture which involves the following components:

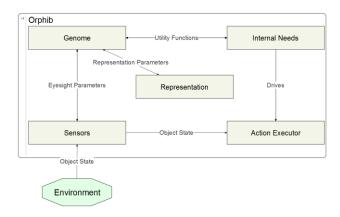


Fig. 5. The agents' architecture diagram.

 Action Executor - Module responsible for making Orphibs deliberate and execute actions (described in Reasoning System). It contains all of the available actions as well as a steering behavior mechanism [8], allowing the Orphibs to seek their destinations. Moreover, the Action Executor

- receives information from the Sensor and Internal Needs modules.
- Representation Module containing the Orphibs' visual representation: eye and body pigmentation, animations and the aging mechanism (explained in more detail in subsection Orphibs Mechanics Overview). Physical traits of the Orphibs, such as their eye and body color, are stored in the genome module.
- Sensors Module that perceives the world's objects as well as their external state. In the current version, the Orphibs only possess a visual sensor, i.e. virtual eyes, that varies according to day light intensity.
- Internal Needs A module that stores and updates the satisfaction levels of the Orphibs' needs at a constant rate during every update time frame. Each satisfaction decays linearly according to a percentage with a fixed variable, decayRate: hunger decays with a rate of 1.1 * decayRate, social decays with a rate of 1.7 * decayRate and fun decays with a rate of 0.7 * decayRate. Reproduction and energy decay differently: the former only starts to decay when the Orphib reaches 10% of his total life span, to simulate puberty, with the rate of
 - $4*\frac{maxAge}{1+(age-0.1*maxAge)}*decayRate$. This means that while the Orphib ages, its satisfaction decaying rate decreases. On the other hand, energy changes according to the expression decayRate*(1.7-eyesight), where eyesight is the total relative eyesight given the Orphibs' eyes reaction to light intensity. This means that when Orphibs see less, they need to spend more energy.
- Genome Module that stores the Orphib's complete genotype that is used during reproduction (for more details seesubsection Evolutionary Mechanism.

Reasoning System. The reasoning system behind the Orphibs is an implementation of the Goal-Based Behavior [8]. More specifically, it is the version present in The Sims 2 game [10]. As such, the algorithm is as follows:

- 1. Iterate through an action queue.
- 2. If there is an action, execute it and get rewarded, if applicable.
 - a. If empty, select a new action:
 - b. Iterate through every world object that can be acted upon.
 - c. Rank each of the object's available actions using a score function.
 - d. Add actions with scores above a given threshold.
 - e. Sort all of the actions by rank.
 - f. Select randomly one of the top 3 actions.

- 3. If still empty execute a random fallback action.
- 4. Repeat step 1.

This implementation takes into account some of the upgrade suggestions presented in [10], namely the fact that step e. was introduced in order to reduce deterministic behavior and how the action rank is calculated. Indeed, the score function used is given by the expression:

score(action) =

$$= \sum_{needs}^{needs} [U_{need}(current) - U_{need}(future)]$$

Where, for a given need, current is the Orphib's current satisfaction level, future is the sum of the current satisfaction level with the object's satisfaction fulfillment and U_{need} is the respective utility function.

Additionally, Orphibs can perform a meta-action "Wait" during the execution of another action. For instance, this happens when Orphibs want to play with toys that are being used by other Orphibs. In this case, the Orphib stops and waits for its turn. However, while waiting, the Orphib checks for better actions, by performing the selection sub-algorithm with a threshold equivalent to the current score of the action it is executing. Moreover, the halted action's score decays 10% every time the Orphib performs an unsuccessful selection. All in all, this decreases deterministic behaviors and reduces the probability of racing conditions.

In order to avoid unexpected behaviors, whenever an Orphib is picked up by the player's hands, it stops its current action and, in addition, aborts the current action of other Orphibs when engaged in group activities, such as talking.

Finally, the Orphibs' reasoning system also implements a basic personality model. This is accomplished by having different utility functions for each need [10], on each Orphib. As previously explained, utility functions are used to score actions on the reasoning system. They map a need value to a value that ranges from 0 to 1. Utility functions in Orphibs are created as data points that describe curves. This was chosen instead of a mathematical expression in order to better support genetic personalities (see subsection Genetic Personalities).

Evolutionary Mechanism

Since Orphibs support reproduction, they have an internal evolutionary mechanism consisting in a simplified Genetic Algorithm (GA) [32]. Nevertheless, some steps of the GA were not considered. For instance, Orphibs do not have an explicit fitness function, instead, their fitness is represented by their ability to survive long enough to reproduce. In addition, parent selection and survivor selection are not performed in a system-centric manner as the literature suggests but rather by the Orphibs' internal reasoning system and maturing mechanism respectively.

Algorithmically, the Orphibs' reproduction works in the following manner:

- 1. Approach potential mate.
- 2. Perform mating dance/animation.
- 3. Create egg.
- 4. Inseminate egg.
- Recombine the genetic material from both progenitors.
- Fulfill the reproduction need of both Orphibs by the same value.

7. Hatch the egg after 10 seconds have passed.

For recombination purposes, it is considered that the Orphib that engages in the reproduction action is the alpha progenitor, while the other is the beta progenitor. It is worth noting that the beta progenitor does not get to choose if he participates in the reproduction or not.

Genome. The Orphibs' genotype is composed of a data structure containing several data types. While it could have been implemented as a floating point vector, using a data structure increased its readability. Therefore, the Orphibs' genotype is as following:

- **Maximum Size** Floating point that represents the Orphibs' maximum size.
- **Maximum Age -** Floating point that represents the Orphibs' maximum age.
- Maximum Vision Floating point that represents the farthest an Orphib can see under ideal conditions.
- Minimum Vision Floating point that represents the farthest an Orphib can see under non-ideal conditions.
- **Eye Color** Floating point vector (for RGBA colors) that represents the Orphibs' eye color.
- **Body Color** Floating point vector (for RGBA colors) that represents the Orphibs' body color.
- Personality Vector of data point vectors. This represents the Orphibs' personality.

Additionally, these attributes are grouped so that the recombination operator is applied to each group independently. The groups are: one containing maximum size and maximum age, one composed of maximum vision and minimum vision, one comprising the eye color, one containing the body color and one for the personality.

Genetic Personalities. As mentioned, Orphibs have a simple personality profile given by their utility functions. These utility functions are transmitted from parents to offspring using the variation operators described in the next section. As such, Orphibs feature genetic personalities. Genetic personalities account for an interesting gameplay mechanic because players can use artificial selection to their advantage in order to produce Orphibs that behave the way they like. Artificial selection can be achieved by picking up an Orphib with the need to reproduce and placing it near another Orphib. The potential alpha progenitor will then attempt to reproduce with the closest Orphib in its range.

Variation Operators. The Orphibs' evolutionary mechanism uses both a crossover operator and a mutation operator. Recombination is applied to each group of the genome independently with a probability of 0.99 (although the literature suggests typical values of 0.9, this was not considered a typical application of GA). This means that each group has a 1% change of being cloned from the alpha progenitor's genome. The recombination algorithm used for most groups was Whole Arithmetic Recombination [32] with an alpha factor of 0.5 while the personality group uses an Uniform Crossover [32] with a p value of 0.5.

Unlike crossover, mutation is applied to each attribute, with a probability of 0.3 (this value was chosen in order to provide some local variability). The operator used was Non-Uniform Mutation with a Gaussian Distribution [32] with the previous value of the genome attribute as means and standard deviation of 1.

Experiment

Experimental Setup

We conducted an experiment to verify how a population of Orphibs behaves without the player's intervention, including if the population increases or stagnates overtime, if the average lifespan changes or if behaviors emerge.

The experiment consisted in running the Orphibs prototype during a 14-minute run with 15 initial Orphibs, randomly generated. It is worth noting that most of the Orphib's attributes were generated from a set of value ranges. For instance, the maximum size ranged from 0.1 to 1.1; the maximum age ranged from 1 to 6; finally, the eyesight's minimum and maximum values ranged from 0.5 to 10.5. These ranges were chosen to avoid unexpected events such as, for example, Orphibs dying during initialization due to a life span of 0. However, attributes such as the body and eye colors and the utility functions were generated without any additional boundaries, besides the ones imposed by their domains (for example, colors could only be generated inside the values permitted by the RGB model)

During the experiment, we recorded the population's size and the Orphibs' average age and maximum life span. This gives an idea of how the population evolves during the experiment. Other attributes such as average color, size, etc., were discarded as they did not have any impact on the population's evolution. In addition, we observed the Orphibs in order to access if any kind of behaviors emerged.

Experimental Results and Discussion

Figure 6 illustrates the Orphib population's evolution during the test:

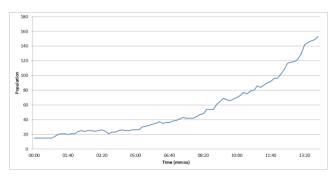


Fig. 6. Orphibs' population evolution over time during the test run.

As can be observed, the population tends to increase nonlinearly without signs of stabilizing. In fact, during the tests it was observed that Orphibs developed a compulsive reproductive behavior, even though the reproduction's satisfaction levels' decaying rate decreases as the Orphibs age. Indeed, by the time the test ended, most Orphibs presented a reproduction utility function close to 1. This means that regardless of the reproduction need, its utility would always be high.

Another factor contributing for this increase is that, according to the Orphibs' death probability function, Orphibs are most likely to die of old age than from other events (The Future Work section presents some suggestions to circumvent this).

Figure 7 presents the Orphibs' average age and respective standard deviation:

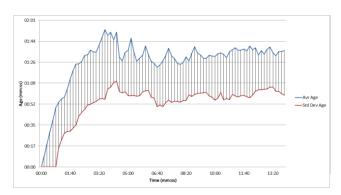


Fig. 7. The population's average age and respective standard deviation, expressed in minutes, during the test run.

There is a time slice, from 00:00 to roughly 05:00, where the average age continuously increases. This can be explained by the fact that, during this time, the population consists of the initial Orphibs. However, after 05:00, the average age tends to fluctuate between 01:26 and 01:44. Nevertheless, its standard deviation ranges between 00:52 and 01:09, meaning that there is some variety in the population's age.

The Orphibs' population average maximum life span is presented in the next figure:

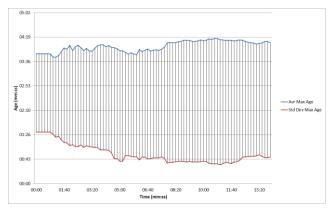


Fig. 8. The population's maximum average life span and respective standard deviation, expressed in minutes, during the test run.

Unlike the population's average age, its maximum life span is more stable: while the average ranges from 03:36 to 04:19, becoming closer to 04:19 at roughly 08:20, the standard deviation decreases to values less than 1 minute.

From the results in Figure 7 and Figure 8 we can conclude that, on average, the Orphibs were close to their midlife and, consequently, above the age where they start to have the need to reproduce. In fact, this corroborates the results in Figure 6

as there are no strong indications that the population is aging. Therefore, it would likely continue to increase in size.

Future Work

As was observed in the experiment, the Orphibs' population increased non-linearly. This was mostly caused by an insufficient population control mechanism. Suggestions of population control mechanism include the manipulation of the Orphibs' internal mechanisms or even the environment.

One of the most basic solutions, is to tweak the death probability function, either by changing the probability curves or by attributing more weight to the energy. Additionally, this function could be augmented to include hunger as a factor. However, balancing the function in order to stabilize the population can be difficult to achieve as reproduction is also dependent on the Orphibs' reasoning system.

Another method is to implement a choosing mechanism for mating. Currently, there is no potential mate criteria, unless a mate is provided by the player. By introducing some criterion that, for instance, restricts which Orphibs can be used for mating another one, reproduction may be more controlled.

Manipulating the environment to control the Orphibs everincreasing population can be done by introducing the concept of predators, however this is beyond the scope of the prototype. Another way, is to introduce resource scarceness.

In addition to the system described in this paper, there were some planned features, such as implementing an emotional model, which, we believe, are interesting for future research but that, due to time constraints, had to be cut from the prototype.

Currently, the Orphibs are omniscient. This means that they have all of the information regarding the world state. However, initially, they were planned to have limited knowledge, and would use their eye sensors to discover landmarks and items. Additionally, they would talk with other Orphibs to swap information about the world. Moreover, they would use their memory to find objects. This could be implemented using the architecture described in [33] where agents explore the environment in order to fulfil their desire to minimize hunger, maximize information gain and maximize surprise

Another feature that was not implemented in the prototype was emotions. Implementing computational emotional models such as the Computational Belief-Desire Theory of Emotion (CBDTE) [34] could be achieved by combining the Orphibs' reasoning system with a memory mechanism. More specifically, the Orphibs' knowledge of the world could be converted into beliefs while the actions to be performed could be converted into desires. This way, beliefs and desires could be fed to the CBDTE in order to produce emotional responses. Consequently, the emotions could be used to influence actions.

Finally, the DNA mechanism could be improved further. It was planned to contain additional physical morphology traits such as body parts' textures. Computationally this would translate to texture morphing. In addition, the prototype was planned to contain the concept of species, i.e. Orphibs whose DNA differ from each other above a given threshold. Specie classification was to be implemented using a clustering algorithm. Moreover, it was thought to be introduced as a

gameplay mechanic: the player would increase a score by creating and maintaining different species.

Conclusion

We presented an overview and agent architecture of the A-Life game prototype we developed: Orphibs II. As a game, Orphibs provides gameplay mechanics present in other A-Life games. However, while the prototype draws some inspiration from other A-Life games, such as The Sims and Creatures, Orphibs II introduced a mixing of concepts which originated in genetic personalities: personalities that are transmitted from parents to offspring.

We also performed a 14-minute test run where 15 randomly generated Orphibs populated the game world. Results indicated that the population tended to increase non-linearly as a compulsive reproductive behavior emerged in the population, during the experiment. Nonetheless, we presented, as future work, several means to potentially solve this problem and provide some population stabilization.

Acknowledgements. This work was supported by the CISUC ICIS project CENTRO-07-0224-FEDER-002003.

References

- Rollings, A. and Adams, E. (2006). Artificial Life, Puzzle Games, and Other Genres. Fundamentals of Game Design. Prentice Hall, New Jersey, pages 477-498.
- [2] Grand, S., Cliff, D. and Malhotra, A. (1996). Creatures: Artificial Life Autonomous Software Agents for Home Entertainment. Autonomous Agents and Multi-Agent Systems, 1(1):39-57.
- [3] Maxis (2000). The Sims, Electronic Arts.
- [4] Barreto, N. (2013). A Language for Game Design and Choreography, MSc. thesis, Department of Informatics Engineering, University of Coimbra.
- [5] Epstein, J. M. and Axtell, R. (1996). Growing Artificial Societies: Social Science from the Bottom Up. Brookings Institution Press, Washington.
- [6] Monteiro, I. M. and Alvares, L. O. (2009). Uma Infraestrutura de Teamwork para Ambientes Dinâmicos. In VII Encontro Nacional de Inteligência Artificial, Rio Grande do Sul.
- [7] Ocio, S. and Brugos, J. A. L. (2009). Multi-agent Systems and Sandbox Games. In AISB 2009, Edinburgh.
- [8] Millington, I. and Funge, J. (2009). Artificial Intelligence for Games, Burlington. Morgan Kaufmann Publishers.
- [9] Wooldridge, M. (2011). An Introduction to MultiAgent Systems. Wiley, Glasgow.
- [10] Zubek, R. (2010). Needs-Based AI. Game Programming Gems 8. Course Technology PTR, pages 302-311.
- [11] T. Baptista and E. Costa, "Evolution of a multi-agent system in a cyclical environment," *Theory in Biosciences*, vol. 127, no. 2, pp. 141-148, 2008.
- [12] Aylett, R. and Cavazza, M. (2001). Intelligent Virtual Environments - A State-of-the-art. In Eurographics 2001, STAR Reports, pages 87-109.
- [13] Gardner, M. (1970). Mathematical Games: The fantastic combinations of John Conway's new solitaire game

- "life". Scientific American, (223):120-123.
- [14] Bach, J. (2009). Principles of Synthetic Intelligence. PSI: An Architecture of Motivated Cognition. Oxford University Press, Oxford.
- [15] Lim, M. Y., Dias, J., Aylett, R. and Paiva, A. (2009). Intelligent NPCs for Educational Role Play. In Agents for Games and Simulations: Trends in Techniques, Concepts and Design, pages 107-118. Springer, Berlin.
- [16] Epic Games (2004). Unreal Tournament. Atari.
- [17] Köster, M., Novák, P., Mainzer, D. and Fuhrmann, B. (2009). Two Case Studies for Jazzyk BSM. In Agents for Games and Simulations: Trends in Techniques, Concepts and Design, pages 33-47. Springer, Berlin.
- [18] Hindriks, K. V., van Riemsdijk, B., Behrens, T., Korstanje, R., Kraayenbrink, N., Pasman, W. and de Rijk, L. (2011). Unreal GoalBots: Conceptual Design of a Reusable Interface. In Agents for Games and Simulations II: Trends in Techniques, Concepts and Design, pages 1-18. Springer, Berlin.
- [19] Sims, K. (1994). Evolving 3D Morphology and Behavior by Competition *Artificial Life*, 1(4):353-372.
- [20] Graham, L. (2007). 3D Virtual Creatures Evolution.
- [21] Dubrofsky, E. (2008). Evolution Mutates Beyond Biology. [Online]. Available: http://www.cs.ubc.ca/~dubroe/journalism/evolution.pdf. [Accessed 21 October 2013].
- [22] Millennium Interactive (1996). Creatures, Mindscape.
- [23] Grand, S. (2013). Grandroids.
- [24] Activision (1985). Little Computer People, Activision.
- [25] PF Magic (1995). Petz, PF Magic.
- [26] Frank, A., Stern, A. and Resner, B. (1997). Socially Intelligent Virtual Petz. In Socially Intelligent Agents AAAI Fall Symposium, Cambridge.
- [27] Stern, A. (2003). Virtual Babyz: Believable agents with Narrative Intelligence. In *Narrative Intelligence*, pages 215-227. John Benjamins Publishing Company, Amsterdam.
- [28] Sonic Team (1998). Sonic Adventure, Sega.
- [29] Sonic Team (2001). Sonic Adventure 2, Sega.
- [30] Schumacher, J. (2013). Species: Artificial Life, Real Evolution.
- [31] Unity Technologies (2013). Unity3D, Unity Technologies.
- [32] Eiben, A. E. and Smith, J. E. (2007). Introduction to Evolutionary Computing, Springer, Berlin.
- [33] Macedo, L. (2006). The Exploration of Unknown Environments by Affective Agents. Ph.D. thesis, Department of Informatics Engineering, University of Coimbra.
- [34] Reisenzein, R. (2009). Emotional experience in the computational belief-desire theory of emotion. *Emotion Review*, 1(3):214-222.

Artificial Life and the Philosophy of Science

Mostafa Fahmy

Carnegie Mellon University, Pittsburgh, PA 15213 mfahmy@cmu.edu

Abstract

What is the ontological status of Artificial Life systems? This question admits two interpretations. The first asks whether and to what degree Artificial Life systems could potentially be considered 'alive' in the sense that we take naturally occuring biological systems to be. The second has to do with what sort of role Artificial Life systems occupy in scientific inquiry. In this paper, we investigate both facets of the question, and explore the philosophical and practical implications of possible answers for scientific practice.

Introduction

The contemporary conception of Artificial Life as a discipline was put forward by Christopher Langton. Langton (1993) characterized the field broadly as "devoted to studying the scientific, technological, artistic, philosophical, and social implications" of creating "living" artifacts.

For many of its practitioners, the ultimate goal of Artificial Life is the synthesis of systems that exhibit life-like behavior, organization, and complexity. In engineering and evolving such systems, we hope to gain insight into the dynamics which govern the natural phenomena that inspire them.

Artificial Life raises a number of compelling issues for the philosophy of science. One of the most pertinent concerns can be stated as follows: what is the ontological status of Artificial Life systems?

This question admits two interpretations. The first, a broad interpretation which we shall call *the metaphysical question*, has to do with the place of Artificial Life systems in our ontology of the world at large. Specifically, it asks whether and to what degree Artificial Life systems could potentially be considered 'alive' in the sense that we take naturally occuring biological systems to be. The second, a narrow interpretation which we shall call *the methodological question*, has to do with what sort of role Artificial Life systems occupy in the context of scientific inquiry. Are they theoretical entities, objects of experiment, or something altogether different?

The present paper seeks to explore both interpretations. First, we will trace the development of a particular strain of

ideas from the Western philosophical canon to inform our understanding of the metaphysical question, as well as the roots of Artificial Life as a discipline. Some scientific and ethical implications of potential answers to the metaphysical question will be discussed.

Second, we will survey a variety of perspectives on the role that Artificial Life systems play, or could potentially play, in scientific practice. This portion of our exploration will focus primarily on software-based Artificial Life.

Philosophical Motivations for Artificial Life

Proponents of the Artificial Life (hereafter ALife) program claim that the construction of living or life-like artifacts can help in addressing fundamental problems and open questions in the science and philosophy of biology. It will serve our inquiry to begin by considering some possible motivations for this claim. Three such motivations are discussed below.

The Generalization Problem

Biologists and philosophers of biology are interested in deriving general principles and theories regarding living systems. Langton (1989) argues that this task requires identifying the essential properties of living systems and separating them from those that are accidental.

A fundamental obstacle to this endeavor is that all naturally occurring forms of life share a common origin. This makes it difficult, if not impossible, to differentiate between those features which are shared among living systems because they are essential, and those that are universal (due to common origin) but accidental. This presents a problem for anyone wishing to form general theories about life which capture only its essential properties.

One way of overcoming this predicament is to search the cosmos for extraterrestrial forms of life (Smith, 1992). Another option is to attempt to construct new forms of life ourselves. In doing so, we would effectively be exploring "the biology of possible life" (Langton, 1989):

Only when we are able to view *life-as-we-know-it* in the larger context of *life-as-it-could-be* will we really

understand the nature of the beast.

Langton's argument is based on an Aristotelean conception of science, one which entails a metaphysical committed to the existence of one or more *natural kinds* - natural (i.e. human-independent) groupings of particular objects - elements of which are said to share a privileged set of properties (their *essence*) without which they would cease to be objects of that kind. Specifically, Langton assumes that (1) 'life' is a natural kind, and (2) an entity is said to be alive by virtue of possessing a specific set of essential properties shared by all living things.

The commitment to natural kinds is common among scientific realists - those who take science to be representative of, or at least aiming to uncover, the actual workings of the world. It is Langton's second commitment, the commitment to essences, which casts him as an Aristotelean.

For the non-Aristotelean, there are at least two other possible motivations for the ALife program.

Understanding Through Synthesis

Richard Feynman is said to have written, "What I cannot create, I do not understand." (CaltechArchives, 1988).

Scientists and philosophers of biology are interested not only in describing the behavior of living systems, but also in explaining how that behavior is generated. Attempting to synthesize life-like systems provides us with an opportunity to test our understanding of the dynamics that account for the behavior we observe in living systems.

Overcoming Practical Limitations

Occasionally, there exist real-world obstacles which preclude scientists from carrying out certain types of inquiries. These obstacles may be related to limited budgets, technological constraints, or ethical reservations.

In such situations, computer modeling and simulation can offer an attractive alternative to traditional experimental methods. In the context of biology, this often results in the construction of systems which fall under the purview of Artificial Life.

The Metaphysical Question

There are two primary theses regarding the metaphysical status of artificial life (hereafter, ALife) systems:

The *Strong ALife* thesis is the claim that life is a process which can be abstracted away from any particular medium.

The *Weak ALife* thesis denies the possibility of creating living artifacts, but claims that the mechanisms and dynamics which govern living systems can be studied by constructing artifacts which model or simulate such systems.

There is also a third, more pragmatically oriented thesis, which is orthogonal to the Strong/Weak ALife debate. *Instrumental ALife* simply states that ALife systems can serve as tools for solving practical problems.

The Strong ALife and Weak ALife positions parallel the Strong AI and Weak AI theses, respectively. In artificial intelligence and computational psychology, Strong AI is the claim that it is possible to program computers so that they exhibit general intelligence of the same or similar kind that humans possess. In other words, "the appropriately programmed computer is [a] mind and has intentions" (Searle, 1980). Conversely, Weak AI denies the possibility that computers can think, but claims that they can be programmed to act as though they were intelligent in narrow scopes.

Proponents of Strong AI often ground their position in a functionalist theory of mind. On this view, it is not the underlying structural economy of parts which accounts for intelligence. Instead, it is the functional organization of those parts that is key. Thus, mental states are multiply realizable—the same mental states that occur in the brain can be instantiated in other media so long as the parts of the system are organized in such a way as to be functionally equivalent.

Similarly, Strong ALife can be seen as advocating a functionalist theory of life. The claim is that non-biochemical systems can be organized in such a way that their behavior is functionally equivalent (or sufficiently similar) to that of naturally occurring living systems. On this view, since the appropriate level of description for the properties of life is functional (and thus divorcable from considerations of medium), it follows that any system which satisfies these functional properties is in fact genuinely alive. Put another way, life itself is multiply realizable.

How did we come to arrive at such a view of life? In the next section of this paper, we will trace the history of philosophical ideas which serve as precursors to the contemporary conception of Artificial Life.

The Philosophical Roots of Artificial Life

Ancient Materialism The *physiologoi* (literally *physical* or *natural philosophers*) were the prototypical scientists of Ancient Greece. They perceived lightning in the sky and saw not the manifestation of Zeus' anger, but a natural process to be studied and understood. Among them was Democritus, one of the early atomists. The atomists held that the universe was composed of indivisible atoms whose movements and interactions took place in an infinite void.

Democritus believed that all living things possessed a *psyche*, or soul, which was the feature that allowed them to carry out their life functions. This psyche was thought to be composed exclusively of fire atoms (Berryman, 2010). As archaic as this idea seems today, it is significant in that Democritus' was one of the first materialist accounts of life. It is reductionist, in that it claimed that the property of life could be reduced to and explained entirely in terms of interactions of matter.

Modern Mechanism In the 17th Century, the scientific revolution was sweeping through Europe. One prominent strain of thought that emerged during this period was the idea that the natural world and its parts could be seen as complex machines, similar to intricate watches and automata that were popular at the time. This idea, that we live in a sort of clockwork universe, was called *mechanism*, or the *mechanical philosophy*.

The French philosopher and mathematician René Descartes partially revived Democritus' idea that life had a material basis. Descartes claimed that animals were nothing more than non-sentient automata (Harrison, 1992). Humans were distinct from other animals in that they had minds, which were constituted of a different substance than matter (a position known as Cartesian dualism). Other animals, however, lacked minds —they were merely physical machines. Thus their behavior could be entirely accounted for by the interaction between their internal parts and the influence of outside forces on those parts. This applies to human bodies as well, insofar as they are physical (though the mind, being nonmaterial, merits different considerations). In his 1664 Treatise on Man, Descartes (1985) describes a hypothetical construct, an artificial man whose internal constitution and outward behavior is meant to mimic our own. Descartes concludes:

I should like you to consider, after this, all the functions I have ascribed to this machine [...] I should like you to consider that these functions follow from the mere arrangement of the machine's organs every bit as naturally as the movements of a clock or other automaton follow from the arrangement of its counter-weights and wheels.

The English philosopher Thomas Hobbes, one of Descartes' contemporaries, was an even stauncher materialist. Hobbes is primarily known for his contributions to political philosophy and his computational approach to psychology. Hobbes held that the human faculty of reason was nothing but 'reckoning' —that is, calculation or computation. This being the case, the behavior of man could be explained and predicted if the elements of this computation (the dispositions and forces driving man's actions, and their combinations) could be identified. Hobbes employs this mechanistic understanding of human psychology to provide a proto-game-theoretic account of man's transition from the state of nature (man's pre-societal condition) to the formation of a civil state. This account is given in his seminal work, Leviathan. Hobbes (1651) opens with the following observation:

NATURE (the art whereby God hath made and governs the world) is by the art of man, as in many other things, so in this also imitated, that it can make an artificial animal. For seeing life is but a motion of limbs, the beginning whereof is in some principal part within, why may we not say that all automata (engines that move themselves by springs and wheels as doth a watch) have an Artificial Life?

It is interesting to note that Hobbes' introduction mirrors that of Langton, in an updated version of his original *Artificial Life* paper (Langton, 1993):

"Art" + "Life" = Artificial Life: Life made by Man rather than by Nature. Our technological capabilities have brought us to the point where we are on the verge of creating "living" artifacts.

The Darwinian Revolution In *On the Origin of Species*, Charles Darwin (1859) described the process of evolution by natural selection. Darwin argued that the myriad forms of life we observe are not static, but the result of a process of branching evolution taking place over many generations.

Scientists and philosophers are interested in providing explanations which address phenomena at an appropriate level of description. A level of description that is too low results in an explanation that is difficult to work with. One that is too high obscures the insights one might gain from a more appropriate level of description. One of Darwin's revolutionary insights, which Dennett (1995) calls Darwin's Dangerous Idea, is that the algorithmic level is the level that best accounts for the variation and qualities of life we observe.

The Functionalist Turn In the late 1940's, John von Neumann was investigating the possibility of constructing a self-replicating machine. He undertook the construction of such an automata, described kinematically (with reference to aspects of motion apart from considerations of mass and force), as a thought experiment. Levy (1992) recounts the story of the attempt as follows.

Von Neumann posited a creature living on a lake containing the same elements of which the creature was composed. He went on to describe the information- and material-processing parts of the creature and how these could be combined in such a way as to provide structures by which the creature could, by manipulating the available elements in its environment, construct a copy of itself.

This attempt was widely met with skepticism. For though the process von Neumann described was taken to be sound, his peers thought the origin of such an automata to be too opaque. Von Neumann had posited, in their minds, a black box. One of von Neumann's colleagues, a mathematician named Stanislaw Ulam, suggested to von Neumann do away with the material environment which his creature inhabited and instead place it on an abstract grid, wherein its workings could be described functionally. This led to the birth of the cellular automaton and the publication of *The General and Logical Theory of Automata*, in which Von Neumann (1951) stressed the analogies between living organisms and his digital automata.

The Birth of ALife With Democritus, we see one of the first materialist accounts of life. Centuries later, his ideas are picked up and extended by mechanists like Hobbes and Descartes. They argue that living things are physical automata whose overall behavior is explained by the interactions of their constituent parts and external physical forces. Darwin provides us with an account of life in which the algorithmic processes of evolution by natural selection plays a fundamental role in shaping the variety of mechanisms we observe in living things. In the 20th century, von Neumann abstracts the process of life away from considerations of underlying medium. The character of the narrative is one that begins with a material reductionism of life, which is then given a mechanistic treatment, and ultimately abstracted to yield a functionalist account of life.

This pattern of ideas culminates in the organization of the first Artificial Life conference by Christopher Langton, who also publishes the manifesto that lays out the vision of the ALife program (Langton, 1989). Langton's conception of life, and of Strong ALife, is one that is both "unashamedly functionalist" and "unabashedly reductionist" (Kember, 2013).

Implications of the Metaphysical Question

Foundations of Biology Herbert Simon (1990) writes that the fundamental goal of the sciences is to find invariants. Not all invariants, however, are created equal. Some differ in their scope or degree of generality. Some fields have strict, quantitative, general invariants. Meanwhile, others have invariants that can be described as near, partial, or approximate.

A particularly important class of invariants are called 'laws of qualitative structure' (Newell and Simon, 1976). These laws say something about the essential nature of the systems to which they apply, but the statements they make are often qualitative, rather than quantitative, in nature. Though seemingly simple, Simon argues that such laws are among the most important scientific discoveries that we have made.

One, perhaps *the*, foundational law of qualitative structure in biology is the Cell Doctrine. It states that (biochemical) cells are the building blocks of all living organisms. In other words, the cell is the smallest unit which satisfies the minimal structure and function necessary to be called life.

The functionalist position of Strong ALife stands in stark opposition to the Cell Doctrine. In practice, very few ALife practitioners claim to have realized Strong ALife, and so it is not the case that an alternative minimal unit of life has been proposed. Nontheless, there remains an in-principle objection to the Cell Doctrine from the standpoint of Strong ALife. The Cell Doctrine presupposes that life is necessarily biochemical in nature. Meanwhile, Strong ALife argues that the medium of realization is incidental rather than essential. On this view, it is the functional organization of components,

not their underlying structural economy, which gives rise to the property of life.

Ethics If we are to take the possibility of Strong ALife seriously, then we must ask ourselves whether and to what degree living artifacts would be worthy of moral consideration. This complicates the already difficult question: in virtue of what are living things worthy of moral consideration?

Even if we reject Strong ALife or abstain from judgement, we must, from a practical perspective, consider the issue of whether there are contexts in which the use of Instrumental ALife might be morally problematic. That the reasons underlying the behavior of complex systems are not always fully understood means that they may behave in unexpected or unpredictable ways. There is then a question about what constitutes an acceptable risk if such systems are are to be deployed in circumstances where human life or critical infrastructure may be at stake. These considerations merit further examination, but such a discussion is beyond the scope of the present paper.

The Methodological Question

We now come to consider the role of ALife systems in the ontology of scientific practice. In particular, we will focus on ALife systems which are software-based, also known as *Soft ALife* (Bedau and ProtoLife, 2006).

Three Perspectives

Most accounts of the role of ALife systems in scientific practice broadly fall into one of three categories, which we shall term *abductionism*, *digital naturalism*, and *simulationism*.

Abductionism treats ALife systems as tools that aid in theory construction and conceptual exploration. This category includes accounts such as that of Dennett (1994), who argues that ALife systems can be construed as a form of "prosthetically controlled thought experiments". In these accounts, ALife systems are primarily thought of as theoretical models, which may be implemented in simulation. These models may serve to prime intuitions and generate hypotheses regarding living systems, which may later be tested by more traditional means. On this view, the modeling and simulation of ALife systems may best be thought of as a theory building paradigm (Diallo et al., 2013).

Digital naturalism views ALife systems as source of experimental data which can be used to settle empirical questions. This category includes Langton's conception of ALife systems as actual instances of life, which may be used in drawing conclusions about living systems. In general, digital naturalists perceive ALife systems as bona fide examples of the phenomena which they attempt to capture. Thomas Ray's *Tierra* (Ray, 1991) is an example of ALife research from the perspective of a digital naturalist.

Simulationism, or pseudo-empiricism regards the study of ALife systems as having a sort of empirical 'flavor', but does

not perceive such systems as direct sources of experimental data. On this view, ALife systems may be perceived as rich 'artificial worlds' worthy of scientific investigation in their own right (Bullock, 2014). They may provide insights into the dynamics which govern living systems, but do not themselves qualify as experimental systems from which to draw conclusions about the living systems themselves.

Here it will be pertinent to note that one's stance on the metaphysical question has the potential to influence the way in which one engages the methodological question. In the context of Soft ALife, the distinction between Strong ALife and Weak ALife can be viewed as that between computer realization and computer simulation of life. If one subscribes to the Strong ALife point of view, then digital naturalism is likely to be an appealing position. The construction and study of sufficiently advanced ALife systems would, in principle, be ontologically equivalent to traditional experimentation.

Implications of the Methodological Question

The methodological question is intertwined with a number of epistemic concerns regarding ALife research. We are interested in ascertaining character of the knowledge we acquire from building and studying ALife systems. Specifically, we wish to know

- 1. What is it that we actually learn *about* from pursuing such a course of inquiry?
- 2. How can we trust that our constructions accurately instantiate or reflect the phenomena they attempt to capture?
- 3. What justifies the inferences we make from the behavior of these systems to the behavior of living systems at large?

Knowledge About What? For the abductionist, any knowledge gained about an ALife system is strictly theoretical knowledge about the dynamics of a particular model being proposed or examined. Though the ultimate aim of the model may be to describe the behavior of some particular class of living systems, it is the model itself and not the target system which is the object of study.

The digital naturalist, meanwhile, will claim that the output of their investigation is knowledge about a genuine living artifact in particula. They may argue that this knowledge is applicable to some more general class of living systems as well.

Here the simulationist resides in a camp similar, but not identical, to that of the abductionist. Simulationists generally take themselves to be learning about the organization and behavior of objects in the artificial world or virtual laboratory of their construction. These may resemble naturally occuring systems, but the latter are not the targets of their inquiry.

The Problem of Justification A traditional distinction in philosophy of science divides scientific practice into two distinct contexts. The context of discovery, which includes activities related to theory construction, is not thought to governed by any rules. In generating models and hypotheses to account for phenomena of interest, anything is fair game. On the other hand, in the context of justification, which includes activities related to theory testing and experimentation, the scope of legitimate activities is strictly constrained. This is because it is within the context of justification that scientific beliefs are acquired, so actions performed therein must be epistemologically justified.

Under which context does the construction of ALife systems fall? The abductionist is likely to see ALife systems as belonging to the context of discovery. Consequently, there is no immediate problem of justification for the abductionist. The abductionist's model is a theoretical proposal, amenable to vetting and testing at a later stage of inquiry via other means. Insofar as these systems are and remain theoretical objects, the process of their construction is not constrained in any way.

The digital naturalist claims that their systems are bona fide instances of the phenomena they are interested in studying. If this is indeed the case, then conclusions drawn about these systems ought to be true *prima facie*. The very existence of these systems serves as their own justification.

The case of the simulationist is less clear. On the one hand simulationism makes no claim to empricial content with respect to questions about living systems. At the same time, such inquiries are in some sense experiments that revolve around the contents of the simulation. Simulation in general, and ALife in particular, is a deeply troublesome case for the distinction of contexts proposed above. It may be the case that these systems represent a 'third way' of doing science (Ruas et al., 2011), one which does not map neatly onto either side of the traditional divide.

Given some set of conclusions regarding a particular ALife system, how do we go about translating those conclusions some more general class of living systems?

For the abductionist, the answer is straightforward (at least in theory). The predictions of ALife models are to be tested in the same manner as any other proposed theory within a given domain.

The digital naturalist will argue that conclusions drawn about a particular ALife system are equally applicable to some class of living systems (whether possible or actual). Skeptics will argue that the digital naturalist remains responsible for providing philosophical and scientific grounding for their systems.

The simulationist begins with the premise that the knowledge gained from an ALife inquiry is encapsulated within the scope of the simulation. Extending these conclusions to living systems at large requires exiting this framework and adopting an approach similar to that of the digital naturalist.

References

- Bedau, M. and ProtoLife, S. (2006). Automated design of artificial life forms from scratch. In *Explorations in the Complexity of Possible Life: Abstracting and Synthesizing the Principles of Living Systems: Proceedings of the 7th German Workshop on Artificial Life, July 26-28, 2006, Jena, Germany*, page 6. IOS Press.
- Berryman, S. (2010). Democritus. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Fall 2010 edition.
- Bullock, S. (2014). Levins and the lure of artificial worlds. The Monist.
- CaltechArchives (1988). Richard feynman's blackboard at time of his death. File: 1.10-29.jpg.
- Darwin, C. (1859). On the origins of species by means of natural selection. *London: Murray*.
- Dennett, D. (1994). Artificial life as philosophy. *Artificial Life*, 1(3):291–292.
- Dennett, D. C. (1995). Darwin's dangerous idea. *The Sciences*, 35(3):34–40.
- Descartes, R. (1985). *The Philosophical Writings of Descartes: Volume 1*. Cambridge University Press.
- Diallo, S. Y., Padilla, J. J., Bozkurt, I., and Tolk, A. (2013). Modeling and simulation as a theory building paradigm. In *Ontology, Epistemology, and Teleology for Modeling and Simulation*, pages 193–206. Springer.
- Harrison, P. (1992). Descartes on animals. *Philosophical Quarterly*, 42(167):219–227.
- Hobbes, T. (1651). Leviathan. Dover Publications.
- Kember, S. (2013). Cyberfeminism and artificial life. Routledge.
- Langton, C. G. (1989). Artificial life, santa fe institute studies on the sciences of complexity. In *Proc*, volume 6.
- Langton, C. G. (1993). Artificial life. Appears as.
- Levy, S. (1992). Artificial life: The quest for a new frontier. *New York, New York: Pantheon*.
- Newell, A. and Simon, H. A. (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3):113–126.
- Ray, T. S. (1991). An approach to the synthesis of life.
- Ruas, T. L., Marietto, M. d. G. B., de Moraes, A. F., Batista, R. d. S. F., Heideker, A., and Aguiar, E. (2011). Modeling artificial life through multi-agent based simulation. *Multi-Agent Systems-Modeling, Control, Programming, Simulations and Applications, Intech.*
- Searle, J. R. (1980). Minds, brains, and programs. *Behavioral and brain sciences*, 3(03):417–424.
- Simon, H. A. (1990). Invariants of human behavior. *Annual review of psychology*, 41(1):1–20.
- Smith, J. (1992). Byte-sized evolution. *Nature*, 355(6363):772–773.
- Von Neumann, J. (1951). The general and logical theory of automata. *Cerebral mechanisms in behavior*, pages 1–41.

Complex ALife Simulations: From Generating Life-like Behaviors to Predicting Real-life Phenomena

Orly (Kramash) Stettiner¹

¹Bar-Ilan University, Israel orlyst@netvision.net.il

Abstract

Artificial Life relates to the study of life-like processes or methods for generating behaviors that may be interpreted as life-like. ALife Computer simulations constitute a significant scientific tool for promoting scientific understanding of biological phenomena and complex natural dynamic processes. Significant leaps in computational force and software engineering methodologies now allow the design and development of large-scale biological models, which—when combined with advanced graphics tools—produce realistic biological scenarios. In some cases, these scenarios may reveal new scientific explanations and knowledge about real life phenomena.

Three state-of-the-art simulation projects are examined in the context of the contemporary philosophical debate on the scientific value of simulations, as we demonstrate the ability of some of these simulations to generate new emergent behaviors, making possible the prediction or hypothesis about the equivalent modeled real-life phenomena. Various considerations in the construction process of the simulation and its user-interface allow the formation of new features and unexpected behaviors, which may serve as speculations in the scientific sense and as a basis for further laboratory research.

Extended Abstract

Computer models, whose dynamic application are termed "simulations", constitute a significant scientific tool for promoting scientific understanding of natural phenomena and dynamic processes in diverse disciplines, including biology. The emerging need of culling significant knowledge and insights from vast amounts of empirical data, generated in recent decades about biological molecules and the millions of interactions among them, has generated innovative sophisticated computational methods. Significant leaps in computational force and software engineering methodologies now allow the design and development of large-scale biological models, which— when combined with advanced graphics tools— may produce realistic biological scenarios, that reveal new scientific explanations and knowledge about real life phenomena.

The living biological structures (which are commonly divided into hierarchical levels of organization, from the chemical-genetic level, through the molecular level to the cells, tissues, organs, organisms and on to populations and ecosystems), clearly represent the idea of *Emergence*. Each level represents a whole, whose functional, structural and behavioral properties are derived from properties and

interactions that take place at lower levels of the organization, but cannot be simply explained by them. According to the holistic systematic approach, which underlies modern biological research, only an integrated view of complex entities allows their true understanding, as well as the detection of emergent novel features and behaviors, which cannot be predicted or viewed at lower hierarchical levels. The fact that biological systems are extremely dynamic, whereas data derived from laboratory experiments are mostly static, requires the use of simulations, to merge the data into an integrated whole, in a parallel, multi-layered, dynamic (and often interactive) manner, in order to allow scientists to yield valuable knowledge.

In recent years there has been an awakening among philosophers of science, seeking to clarify the role played by simulations and their epistemological standing within the space defined by theories, models and scientific experiments, as well as the ability of simulations to scientifically explain real-world phenomena (e.g. (Callebaut 2012), (Eckhart 2010), (Lenhard 2007), (Morrison 2009), (Pennock 2007), (Winsberg 2010)). Following this contemporary philosophical debate, the model construction process is carefully examined for three state-of-the-art test cases, in order to determine its ability to generate novel emergent behaviors that make it possible to predict or make solid hypothesis about equivalent real-life phenomena.

The selected test cases are computer simulations of unprecedented scale of complex biological systems, developed by three leading research groups and involved the development of sophisticated computer models and tools.

Reactive Animation (RA). A group of researchers in Weizmann Institute (Israel) developed a computational approach termed Reactive Animation for simulating the developmental and behavioral processes of an organism (and organs within) (Vainas, Harel et al. 2011). The dynamic characteristics of the biological objects are described based on cellular and molecular data collected from real lab experiments. These data are integrated bottom-up by computational tools and methods to create a comprehensive, dynamic, interactive simulation (with front-end animated visualization) of biological systems behavior and development, in which the "simulationist" may intervene on-line and observe in-silico, on the artificial life-like system, the effects of what may be considered as thought experiments.

In particular, the RA system was reported to have revealed several unexpected emergent properties that were not overtly preprogrammed in the molecular and cellular data included during the construction of the simulation (e.g. competition between thymocytes for sites of stimulation in the thymus). These behaviors may be considered as "weakly emergent" phenomena, meaning that they are incompressible and "cannot be derived except by crawling through all of the gory details of the interactions in the causal web" (Bedau 2013). The discoveries consequently alerted biologists and prompted real lab experimentations of phenomena previously unknown. This renewed investigation of the real world is a process which, according to the researchers, highlights the explanatory power and the potential aid to experimentation offered by an animated interactive ALife simulation of complex sets of data (Efroni, Harel et al. 2005). According to the developers, these models enable in-silico experiments at run-time and produce results that are similar to in-vivo experiments and suggest new intriguing hypotheses (Setty, Cohen et al. 2010).

Digital Organisms of Avida. A research group in DevoLab, Michigan, developed the Avida platform, on which populations of digital organisms evolve under various evolutionary conditions, which leads to the emergent formation of complex traits (Lenski, Ofria et al. 2003), (Ofria, Huang et al. 2008). Through a large series of different experimental environment simulations, performs on these digital organisms (which are self-replicating and evolving computer programs), the researchers demonstrated the ability of the simulated population to dynamically and spontaneously develop novel behaviors and capabilities, not programmed for, which may, again, be considered as weakly emergent.

A major goal of these simulations is to test specific hypotheses related to the theory of evolution and population genetics. However, we claim that due to specific issues related to the modeling process and to the extent of idealization, these simulations should be classified as "how possibly" simulations (Craver 2006), models that are not necessarily based on actual real-world knowledge. Despite the life-like behaviors they produce, they are not sufficiently qualified to scientifically explain or predict real-life phenomena.

Blood clotting nanobots. A UK-based research group developed a series of simulations of coagulation processes, as a basis for physical-medical implementation using nano-robots (Polack, Andrews et al. 2010). A major goal of the group was to develop nature-inspired methodologies to engineer complex emergent phenomena with assurance of functionality and safety. Despite the agent-based simulation' impressive results, producing life-like blood clots, we claim that these are merely phenomena-generating simulations, that cannot be used to predict or enable biologists to gain insight about the actual modeled biological system.

Summary. Based on the above-described test cases, a distinction is made between phenomena-generating models and phenomena-inducing models, and it is shown how the development and execution of a simulation can create actual scientific knowledge, that enables scientific explanation of the phenomenon being modeled.

Simulations, it is claimed, have a "life" of their own, characterized by complexity, creativity and dynamic interactive computational power, which exceeds the standard Turing computation. These features, along with various considerations in the construction process of the simulation and its user-interface, allow the formation of new features and unexpected

behaviors, which may serve as speculations in the scientific sense and as a basis for further research. Specifically, the distinction, present in most relevant publications, between bottom-up models and those constructed from the top downwards is, we claim, fictitious. This insight, according to which these two directions merge into a single whole, underlies the ability of simulations to produce emergent phenomena, which involve complex causal relations, upstream and downstream alike. It is the verification and validation processes integrated into the simulations, the usefulness and contribution of the models in scientific discourse and the cumulative reliability of the applied techniques, which ultimately provide simulations their credibility and scientific significance.

References

- Bedau, M. (2013). "Weak Emergence Drives the Science, Epistemology, and Metaphysics of Synthetic Biology." Biological Theory 8(4): 334-345.
- Callebaut, W. (2012). "Scientific perspectivism: A philosopher of science's response to the challenge of big data biology."

 <u>Studies in History and Philosophy of Biological and Biomedical Sciences</u> **43**(1): 69-80.
- Craver, C.F. (2006). "When mechanistic models explain." Synthese 153(3): 355-376.
- Eckhart, A. (2010). Tools or Toys?. Stuttgart, Institute of Philosophy, University of Stuttgart.
- Efroni, S., D. Harel, et al. (2005). "Reactive Animation: Realistic Modeling of Complex Dynamic Systems." Computer **38**:38-47.
- Lenhard, J. (2007). "Computer simulation: The cooperation between experimenting and modeling." <u>Philosophy of Science</u> **74**(2): 176-194.
- Lenski, E. R., C. Ofria, et al. (2003). "The evolotionary origin of complex features." Nature **423**: 139-144.
- Morrison, M. (2009). "Models, measurement and computer simulation: The changing face of experimentation." Philosophical Studies 143(1): 33-57.
- Ofria, C., W. Huang, et al. (2008). "On the Gradual Evolution of Complexity and the Sudden Emergence of Complex Features." Artificial Life **14**(3): 255-263.
- Pennock, R. (2007). "Models, simulations, instantiations and evidence: The case of digital evolution." <u>J. of Experimental and Theoretical AI</u> **19**(1): 29-42.
- Polack, F. A. C., P. S. Andrews, et al. (2010). <u>Reflections on the Simulation of Complex Systems for Science</u>. Engineering of Complex Computer Systems (ICECCS), 2010 15th IEEE International Conference on.
- Setty, Y., I. Cohen, et al. (2010). "Modeling Biology using Generic Reactive Animation." <u>Fundamenta Informaticae</u> **123**: 1-12.
- Vainas, O., D. Harel, et al. (2011). "Reactive animation: From piecemeal experimentation to reactive biological systems." Autoimmunity **44**(4): 1-11.
- Winsberg, E. (2010). <u>Science in the Age of Computer Simulation</u>. Chicago and London, The University of Chicago Press.

Methodologies

Indefinitely Scalable Computing = Artificial Life Engineering

David H. Ackley¹, Trent R. Small¹

¹University of New Mexico, Albuquerque, NM 87131 ackley@cs.unm.edu, tsmall1@unm.edu

Abstract

The traditional CPU/RAM computer architecture is increasingly unscalable, presenting a challenge for the industry—and is too fragile to be securable even at its current scale, presenting a challenge for society as well. This paper argues that new architectures and computational models, designed around software-based artificial life, can offer radical solutions to both problems. The challenge for the soft alife research community is to harness the dynamics of life and complexity in service of robust, scalable computations—and in many ways, we can keep doing what we are doing, if we use *indefinitely scalable* computational models to do so. This paper reviews the argument for robustness in scalability, delivers that challenge to the soft alife community, and summarizes recent progress in architecture and program design for indefinitely scalable computing via artificial life engineering.

The future of software artificial life

Focusing particularly on models that span traditionally separate representational levels, the 'soft alife' (Bedau, 2003) research community has used digital computers to investigate everything from artificial physics and chemistry to artificial biology and ecology. Collectively, the community has sharpened the understanding of life-like systems, deepened the understanding of their range, and offered illuminating examples of their complexities. Alife models and techniques have found applications in contexts like film production (Bajec and Heppner, 2009; Reynolds, 1987) and video games (Grand, 2003, e.g.).

The productivity of the community has been admirable, particularly given its modest size, but we believe a far greater destiny for it lies in store: Software-based artificial life is to be the architectural foundation of truly scalable and robust digital computing. Because, fundamentally, the mechanisms required for robust scalability—to switch energy and perform work, to adapt to local conditions or maintain invariants despite them, to increase parallel processing to suit available resources—are precisely what life does.

Future computer programs will be less like frozen entities chained to the static memory locations where they were loaded, and more like yeasty clusters of digital cells, moving and growing, healing and reproducing, cooperating and competing for computing resources on a vast digital landscape—that will itself be growing and changing, as we build and upgrade it even while it's operating. Any program instance will be finite, but the substrate architecture itself will be *indefinitely scalable*, defined as supporting openended computational growth without requiring substantial re-engineering (Ackley and Cannon, 2011). An indefinitely scalable machine will ultimately provide an aggregate computational power to dwarf our current visions for even high-performance computing.

To get there from here, we need to reveal, remove, and reimplement design elements that block indefinite scalability. Traditional models of computation, as well as important areas of soft alife research, embody several such assumptions. We need to raise awareness of the costs of such designs, and advocate for existing alternatives, as well as develop new ones.

To get there from here, ultimately a significant societal investment will be required, to back an expanding software artificial life community as it fleshes out a body of scientific and engineering knowledge around robust artificial life for computer architecture. There is much to be invented and discovered, but the payoff will be immense: The development of a tough and savvy computing base of great capability—not a system promising freedom from all risk or fault, but a system for which risk management and fault tolerance have always been inescapable parts of life.

To get there from here, we also need to get going. Recently we have presented a framework called *bespoke physics* to ground the effort (Ackley, 2013a); in addition, we have made the case to communities focusing on operating systems (Ackley and Cannon, 2011), spatial computing (Ackley et al., 2013), and general computing (Ackley, 2013b). Here our purpose is to sound a vigorous call to arms and place a challenge before the soft alife community, and to provide an update on our own recent progress in indefinitely scalable computing via artificial life engineering.

In the rest of this section we expand on the twin challenges—scalability and security—now facing tradi-

tional computer architecture. Section 'Beyond serial determinism' proposes an alternative and grounds it briefly in history, then Section 'Challenges to the soft alife community' highlights common architectural assumptions—such as synchronous updating and perfect reliability—that can yield evocative models in the small, but lead to engineering dead ends in the large. Section 'Programming the Movable Feast Machine' reports progress on our tool-building efforts for indefinitely scalable architecture, along with first benchmarks on (finitely) parallel hardware, and then finally, Section 'A call to action' touches on our next steps and appeals to the soft artificial life community for help.

Computer architecture at a crossroads

The rise of digital computing over the last seventy years has been a stunning feat of technological research and development, with revolutionary economic and societal impacts. But recently the growth rate of traditional serial deterministic computing has plateaued, as further clock speed increases consumed exorbitant resources for diminishing returns. Now 'multicore' machines exchange strict serial determinism for a modicum of parallelism, creating some uncertainty about the exact sequencing of operations, while preserving overall input-output determinism for well-formed programs. But even there, the requirement for *cache coherence*—so the architecture presents a unified Random Access Memory to all processors—is demanding increasingly heroic engineering (Xu et al., 2011, e.g.) even when only considering scalability within a single chip.

At the same time, given recent high-profile computer penetrations and security failures (Harding, 2014; Perlroth, 2013, and many others reported and not), there is underappreciated irony in the computing industry's determined preservation of CPU and RAM architectures, which—by fundamental design—are all but impossible to keep secure. Because programs and data can be placed anywhere in RAM, storage location provides precious little clue to the identity or provenance of its content. And *central* processing means that the *same* tiny spots of silicon run everything—whether code of the long-trusted servant or the drive-by scum of the internet.

Undeniably, serial deterministic computing with CPU and RAM has great strengths: It is flexible and efficient, and its behavior can be predicted accurately by chaining simple logical inferences—which programmers do routinely as they imagine execution of their code. But that predictability exists only so long as hardware and software and user all act as anticipated. If anything amiss is detected, *fail-stop* error handling—that is, halting the machine—is the traditional response. It's game over; no further predictions are required. Fail-stop is efficient to implement and tolerable assuming the only unexpected events are rare faults due to random cosmic rays or other blind physical processes, resulting in nothing more than the occasional system crash without last-

ing damage or too much lost work.

However, this only applies to small and isolated systems. By contrast, as the high-performance computing (HPC) community contemplates the move to exascale computers, the cost of using fail-stop to preserve program determinism is increasingly seen as untenable (Cappello et al., 2009). And for today's networked CPU/RAM computers, the unexpected is typically neither rare nor random. As app installs and updates bring ever more software bugs, and ever more value at risk attracts ever more malicious actors, the only safe prediction is that *the first flaw loses the machine* to an attacker's control. A horror movie nightmare, where hearing one senseless incantation causes immediate and enduring loss of volition, is quite literally true in our digital environments.

We are now building millions of computers per week according to that staggeringly fragile blueprint. Hardware switches are packed millions to the square millimeter and controlled by a software house of cards—a rickety skyscraper of cards, lashed together by a single thread of execution. It's a devil's bargain that we have accepted, it seems, because its efficiency and flexibility strengths were immediate while its security and scalability weaknesses have overwhelmed us gradually. Now we're so deeply invested in the architecture that we blame only the imperfect tenants, never the doomed buildings: We blame the programmers with their buggy code and the harried managers shipping it, and the miscreants with their malware and the clueless users clicking it. We have accepted this devil's bargain, it seems, because we thought there was no fundamental alternative, or that any alternative would involve unaffordable exotic hardware and space shuttle-grade software.

Beyond serial determinism

There is another approach to building digital computers, a direction suggested decades ago but still mostly unexplored today, that leads to *robustness* instead of fragility. It is built not on static correctness but on dynamic stability, aimed not at efficient completion but at continuous creation, acting not via local changes to a frozen ocean but via collective stabilization of restless seas. It is neither free from faults nor paralyzed by them, but born to them, expecting and accomodating them—even exploiting them.

The proposal is to extract large quantities of robust, useful computation from vast ecosystems of engineered, software artificial life, running on a bulk digital hardware substrate consisting of interchangeable commodity processing tiles suitably (re)architected for the purpose.

This may sound like outrageous science fiction, but—at least in the architecture discussed below—it depends only on conventional electronics manufacturing and requires no

¹Or, possibly, the *second* flaw, if there's an active 'user' vs 'administrator' distinction.

breakthroughs in materials science or unconventional computing media. Or it may sound outrageously inefficient, but for truly scalable computation, esteem for efficiency must be tempered with respect for robustness. There *will be* undetected faults in memory and processing, and hardware failures too big to mask, and substantial resources going online and off without notice. The system as a whole simply will not have a global start or halt or reset state.

To prosper in such systems, large and long-lived computations will employ strategic redundancy throughout. They will have capabilities for environmental and internal monitoring, as well as for regulation, repair, migration, and opportunistic replication for robustness and performance.

In short: Large computations will become artificial life.

In fundamental ways, the soft alife research community has been exploring in miniature the sorts of issues that the computing industry is destined to encounter in the large—indeed, that it is starting to encounter now. Soft alife should *own* that connection, and if we embrace indefinite scalability in our models, we can.

The proposal is frankly ambitious, but at the same time, in broad strokes it is far from new. Over sixty years ago, von Neumann (1951) predicted that serial deterministic computing—his namesake approach—was destined to change:

Thus the logic of automata will differ from the present system of formal logic in two relevant aspects:

- 1. The actual length of "chains of reasoning," that is, of the chains of operations, will have to be considered.
- The operations of logic...will all have to be treated by procedures which allow exceptions (malfunctions) with low but non-zero probabilities.

In other words, von Neumann argued, both the openendedness of the 'serial' and the perfection of the 'determinism' would have to go. He used a reliability argument and explicitly contrasted fail-stop with the 'hide and heal' error handling of living systems.

At the time, von Neumann guessed it unlikely that computers would grow beyond perhaps tens of thousands of "switching organs"—we would say "logic gates"—using serial determinism. But today's microprocessors—still deterministic, if no longer entirely serial—often sport upwards of a billion gates, putting von Neumann's prediction in the wrong by some six orders of magnitude. Advances in materials, manufacturing, and electronics design have increased gate reliability and reduced costs immensely, and judicious touches of hardware redundancy where needed—for example in error-correcting memories—have, so far, preserved determinism as seen by the application programmer. But we believe application determinism in the large is a lost cause, as suggested by the HPC reference cited above, and it is high time to begin fleshing out alternatives.

The economics and risks of programmability

The universally programmable machine is certainly among the greatest theoretical ideas in all of computer science. Within their physical limits, manufactured digital computers are, in some sense, the most flexible machines possible; that is their glory—and their Achille's heel. There is an inherent coupling between *inflexibility* and security: If some machine simply cannot do something, as a matter of physics, no mode of failure or successful attack can change that.

More flexibility means more accessible actions and thus more risk. For trustworthy performance on high-consequence tasks—automobile engine control, for example—first principles would favor a rigid, special-purpose machine. But the automotive industry—like many others—is moving just the other way, despite the risks, because the economics of programmability is a tidal wave.

A multibillion dollar investment in a new chip fabrication factory today can be attractive because programmability allows one machine design to be used for myriads of purposes—that can be chosen after the hardware is built or sold. Any architecture hoping to supplant CPU and RAM will have to be significantly programmable, or able to generate high volume unit sales some other way.

Though today it is mostly a gleam in the eye, an indefinitely scalable computing architecture, that combines useful programmability with life-like savvy toughness, stands to offer both, and that is our goal.

The Movable Feast Machine

The specific architecture we are developing—called the *Movable Feast Machine* (MFM)—is designed to combine indefinite scalability with robust programmability, and be implementable in mature and cost-effective technologies. Here we describe it only briefly, to ground the discussion in a concrete example, and highlight the challenges those design goals can present for traditional soft alife modeling. Ackley (2013a) offers additional discussion of indefinite scalability, while details of MFM operations are described in (Ackley et al., 2013).

Consider Figure 1, the canonical MFM architecture overview diagram. To a soft alifer's eye, the MFM will most likely read as an artificial chemistry on a cellular automaton, and that is a fair assessment, although the details of both are unusual. The computational state is embodied by 'atoms', which are fixed-length bitvectors existing one per site like a many-valued CA symbol.

An MFM program consists of a finite set of 'elements' and a finite 'Garden of Eden' state. An element definition is analogous to a class definition in an object-oriented system, with the atoms representing object instances. The element definition specifies the interpretation of the bitvector of its atoms—its 'data members'—and provides a 'behavior' method specifying how type of atom reacts with its neighborhood when an asynchronous state transition (an 'event

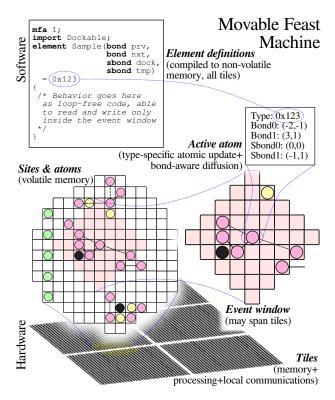


Figure 1: MFM architectural overview.

window' in Figure 1) occurs.

The MFM is unusual as a CA for being asynchronous and employing a relatively huge neighborhood, and its chemistry is entirely discrete, with its reaction rules defined not by transition tables or simple mathematical expressions, but a full-fledged though stylized programming language (originally based on Java, now C++). To enhance robustness, the MFM employs a strict Harvard architecture, placing executable element code in an entirely separate address space from the atoms, so *nothing* that happens during atomic level computations can affect the 'laws of physics' determined by the elements. A separate communications channel, inaccessible to the atoms, is presumed if it is necessary to perform 'magic'—to change or update the compiled element codes (see Ackley et al. (2013) for more).

Challenges to the soft alife community

Several of the MFM architectural decisions—such as the asynchronous updates, and the limited state per atom—are undeniably aggravating for programmers, compared to the heady freedom of serial determinism on CPU and RAM, so it is important to highlight the costs of succumbing to the latter, and the benefits of soldiering on with the MFM. Software-based artificial life, broadly speaking, can be constructed and construed either as 'weak alife'—as life-like computations intended for pedagogy or entertainment or scientific modeling—or as 'strong alife'—as forms of *living*

technology (Bedau et al., 2010) deployable to advantage in computer systems other than just the experimenter's.

On the one hand, 'soft weak alife' can employ whatever architectural assumptions its designer wishes, since the ultimate worth of the computation depends only on properties of its output—being instructive, say, or enjoyable or predictive—and not on the computation's internal structures. But on the other hand, once a computational model exists, it is almost impossible to resist imagining it 'writ large': as a scaled-out system capable of actually *doing something*. In other words, 'soft weak alife' contains an implicit affordance to be interpreted as 'soft strong alife'—and that's when its architectural assumptions become crucial.

The game of Life (Gardner, 1970), as likely the best-known instance of a synchronous, faultless cellular automata, provides a good example of the issues and risks. Explicitly positioned as a game, both fun and instructive, its alluringly simple rules and complex resulting behaviors have inspired generations and spawned an entire subculture of obsessed aficionados. And that is vastly more impact than most simple models can claim, but despite its name and its mathematical and metaphorical appeal, the game of Life is a poor architecture for soft strong alife. Concrete evidence for this can be seen in a recent masterful analysis of the game of Life glider, in which Beer (2014) reports that, of the 2²⁴ possible states of the sites surrounding a glider, an occurence of any of more than 99.44% of those states destroys the glider. Now that's fragile.

Robust systems and actual living systems have *structural degeneracy*—meaning multiple possible states that can perform the same function, allowing them to ride out many environmental perturbations. As alife researchers, we should want to emphasize architectures and models offering massive degeneracy—and yet, as Beer (2014) notes dryly, "Of course, the structural degeneracy of a glider is quite mild, but this degeneracy can be astronomical in more complicated entities." But in such a synchronous, faultless, deterministic model, everything is predictable in principle, so who needs degeneracy? It's inefficient! We could simulate a bigger model instance if we ditched it! Our architectural assumptions shape our models, more often and more deeply than we may recognize.

Here are five architectural properties—of which the first three, at least, are common in soft alife—that are incompatible with indefinite scalability, often for overlapping reasons:

 Global synchronization. Architectures based on a global clock, or presuming global simultaneous updating, are not indefinitely scalable, requiring either globally flawless execution, faster than light communication, or both. Note the oft-rediscovered trick for overlaying a synchronous CA atop an asynchronous one (Nakamura, 1974; Nehaniv, 2004) presumes perfect reliability—if there's one 'stuck' asynchronous site anywhere, the entire synchronous universe grinds to a halt.

- Perfect reliability. Architectures based on globally flawless execution are not indefinitely scalable, as discussed earlier. Our soft alife models, ideally, should be selfstabilizing (Dijkstra, 1974; Schneider, 1993), but at the very least, some non-zero level of random bit corruptions should be tolerable indefinitely.
- 3. Free communication. Architectures based on fixed or sublinear latency in global communications are not indefinitely scalable. Physical machinery occupies space and the speed of light is finite, so global communications latency can be no better than linear in the machine diameter. This also rules out tree-based logarithmic communications latencies, and, perhaps less obviously, any local use of (finitely-old) global data—such as conditioning local reproduction decisions on the current global population size, for example.
- 4. *Excess dimensionality*. Architectures postulating more than three scalable dimensions are not indefinitely scalable, though any number of *finite* additional dimensions is implementable.
- 5. Periodic boundary conditions in three dimensions. Architectures postulating 'wrap around' edges in three scalable dimensions are not indefinitely scalable. In general, edge cases cannot be completely avoided—they can occur due to localized hardware failures, for example—but periodic boundary conditions can be tolerated in models with at most two scalable dimensions.

Again, it is important to stress that by evaluating a soft weak alife model for adequacy as a soft strong alife model, we are going beyond what most designers are prepared to claim for their models—so the blame for any shortcomings, strictly speaking, is on us. But that is not a complete excuse for the model designer, because fragile and unscalable soft weak alife models can tend to mislead our intuitions about the scope and complexity of the rules we should *actually* be considering to make substantial progress. And fundamentally, if one seeks potential designs for soft strong alife, where else should one turn?

There has been an asymmetry in the triumvirate of 'hard', 'soft', and 'wet' alife (Bedau, 2003). The hard alife folk, working in robotics, are obviously constrained by the physical—everything from backlash to friction to irreproducible results. The wet alife folk are likewise constrained by the physical—everything from toxicity to contamination to irreproducible results. Only the soft alife folk were apparently unfettered, but our claim is that, in reality, they too are constrained by the physical, and that indefinite scalability is a suitably abstract but effective way to recognize that.

Programming the Movable Feast Machine

The MFM is an indefinitely scalable architecture designed for research and exploration, and to be readily implementable on low-cost hardware. Our hope is that imple-

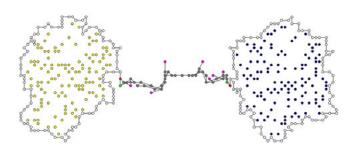


Figure 2: 'Cells' exchanging messages via double-bonded self-healing wire. From (Ackley and Cannon, 2011).

Predicted event rate vs site position 10K avg instr/event, 2Mbps, P1 Atom, N=32, R=4

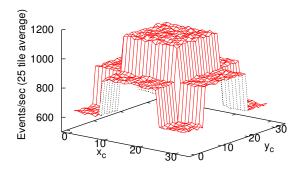


Figure 3: Estimated indefinitely scalable event rate by site, from (Ackley et al., 2013). Compare to Figure 4.

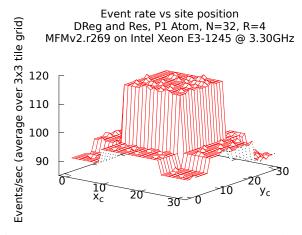
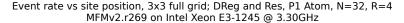


Figure 4: Measured (non-scalable) average event rate by site on DReg. Nine tile simulation running on a 4 core/8 hyperthread processor. Compare to Figure 3 and Figure 5.

mentors of artificial chemistries (Dittrich et al., 2001, is a survey) as well as other alife researchers and interested programmers in general, ultimately, can be enticed to work with it. For that to happen, we need to make it readily accessible to all, and that is the goal of our current development work,



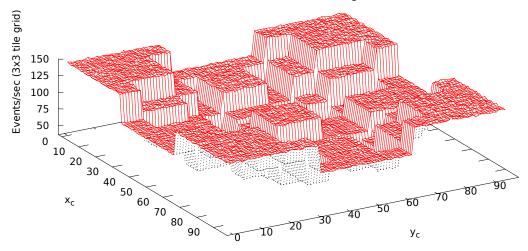


Figure 5: Measured event rates over all sites in a 3×3 grid. Same simulation as in Figure 4.

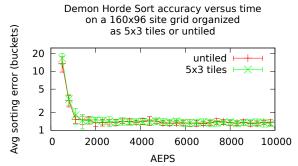


Figure 6: Average sorting accuracy of Demon Horde Sort in a monolithic space vs a 5×3 grid of tiles. See text.

which we describe briefly here.

An MFM simulator and operating system

Most MFM simulation results reported in Ackley et al. (2013) and prior—such as the 'communicating cells' model in Figure 2—were obtained with a Java simulator that acted like one big tile. We are now developing a C++ 'MFMv2' codebase, aimed at using the same core event processing code both in simulators running on traditional machines—using one thread per simulated tile—and eventually, in the operating system for actual indefinitely scalable hardware tiles. With this approach, a model built on the finitely-scalable threaded simulator will likely require only a recompilation to run very similarly on indefinitely scalable hardware—and while in simulation, we have easy global visibility into all tiles for debugging and data gathering.

To aid programmability, the MFM architecture employs intertile locking to make single event execution effectively serial deterministic even if the event window spans tiles. The studies in Ackley et al. (2013) predicted such locking would depress event rates at the edges and corners of tiles, as depicted in Figure 3. In our early experiments with the new multithreaded simulator, we have indeed observed those effects, as seen in Figure 4, which illustrates the average events per second at each site of a tile, averaging over a 3×3 grid of tiles. On average, we observe that events happen about 30% more often in the central 'hidden' regions requiring no communication with surrounding tiles, compared to the edges and especially the corners. However, Figure 5 shows that this 'average tile' is a mixture of two very different situations—central and edge—and the edge tiles actually see significantly higher event rates due to decreased locking.

Such variability in event rate may seem positively debilitating, coming from traditional models in which we are expected to orchestrate the motion of every bit, but indefinite scalability requires the software as well as the hardware to be robust. For example, the DReg physics used in the benchmark above (and as a part of the 'QBar' simulation below), causes random atoms to be deleted every so often, so any computation sharing space with DReg must be robust to that. It is possible to consider mechanisms to level the event rate across tile sites, but they will incur a possibly significant performance penalty, and we may find that this hardware-induced event rate variability is better taken as just another factor to which the software stack will adapt.

For example, we have performed a number of studies on a robust sorting algorithm we call the 'Demon Horde Sort', in which Sorter atoms move Datum atoms across the grid while sorting them vertically by value. This algorithm is described in greater detail in Ackley et al. (2013), but its reported behavior there was based on single-threaded simu-

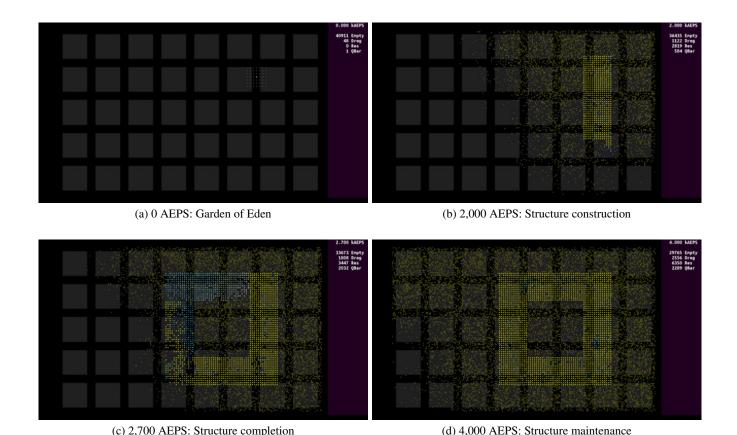


Figure 7: Stages in building a large structure, showing recycling atoms (Dreg, grey), resource atoms (Res, brown), and structure atoms (QBar) both immature (from white to blue) and mature (yellow). Assembly begins slowly (a), (b) but completes rapidly (c) and is maintained indefinitely (d) as the system equilibrates. The tile structure is visible in the background (grey).

lations. Considering the wide variation of site event rates in these tiles, compared to a single-threaded model, we wondered how much this would affect the Demon Horde's performance. Figure 6 shows early results.²

(c) 2,700 AEPS: Structure completion

We ran two simulations of the same dimensions and total sites: One on a untiled grid, 'untiled', and another on a grid of 5×3 tiles, 'tiled'. The untiled simulation sees a uniform average event rate at all sites, while the tiled simulation experiences the variable event rates illustrated in Figure 4 and yet there are no obvious differences in sorting accuracy between them. Much more experimentation is needed, but this does suggest it is possible to design useful computations despite significant variabilities in the neutral dynamics of the architecture—the spontaneous underlying system behavior independent of any particular program.

Higher order structures

The MFM's explicit programmability makes it relatively easy to create a range of specific collective behaviors, even when the resulting structures are much larger than one atomic neighborhood (the event window in Figure 1). As an example, Figure 7 displays screenshots from a simulation that builds a four-sided structure of 'QBar' atoms, measuring over 100 sites on a side, spread over 20 tiles. Simulated time is measured in average events per site (AEPS).

The QBar atom uses bits of its atomic state to represent:

- The overall width (5 bits) and height (7 bits) of the 'bar' it is part of,
- Its x_r (5 bits) and y_r (7 bits) relative position in its bar,
- Its symmetry (2 bits) in terms of 90° bar rotations, and
- Its maturity represented as a 3 bit stochastic counter.

The QBar transition rule performs several actions, including consistency checks involving neighboring QBar's, and seeking out available resources (Res atoms) to transmute into further QBar's, if a gap is found in the structure. Also, QBar may decay itself back into a Res, if it finds itself highly inconsistent with its neighborhood, and as a special case, when a QBar atom matures in the maximum position (i.e., when $x_r = width - 1$, $y_r = height - 1$, and

²Note the buckets are somewhat coarser in this case compared to prior reports, so the average bucket errors are not directly comparable across implementations.

maturity = 7), it seeks to create a minimum position QBar atom 'around the corner' using the next rotational symmetry.

The resulting <code>QBar</code> 'box' is not particularly useful on its own; it is shown here as an illustration of the kinds of structures that are relatively easy to build in the MFM. Extensions to <code>QBar</code> can be readily imagined, such as using the empty sites in its deliberately porous structure to actively transport other selected elements across the bars, generating distinct inner and outer chemical environments. In general, as experience grows with the MFM, we hope to build up libraries of elements and their components—<code>quarks</code>—to make bespoke physics design and construction easier and more modular.

A call to action

A major goal of the MFMv2 second generation simulator is to make exploring robust computations with the MFM—for alifers and programmers at least—as painless as possible, to help stimulate the growth of a community and a body of science and engineering knowledge around indefinitely scalable computing via artificial life engineering.

As we finalize this paper for the proceedings, the simulator is still under heavy development, but by the date of the conference we are hoping to have a version 1.0 release, complete with documentation, tutorials and Ubuntu packaging so that installation can be as simple as apt-get install mfmv2 from a public personal package archive.

But it is not necessary to use our software, or even the MFM architecture itself, to contribute to the larger effort to develop robust-first computing and flesh out alternatives to traditional serial deterministic computing. The power of indefinite scalability, if we define and refine it properly, is that any indefinitely scalable architecture will be transformable into any other of equal or greater scalable dimensionality, at only plausible cost. Or, in particular, transformed into actual hardware at potentially massive scales.

For too long, soft alife models have been separated from each other by their unique laws of physics. But there can be strength in unity. Join us.

Acknowledgments

This work was supported in part by a Google Faculty Research Award, and in part by grant VSUNM201401 from VanDyke Software, both to the first author. Lance R. Williams, Thomas P. Jones, G. Matthew Fricke, and the Robust-First Computing Group at UNM contributed valuable ideas and a place for them to grow.

References

- Ackley, D. H. (2013a). Bespoke physics for living technology. *Artificial Life*, 19(3-4):347–364.
- Ackley, D. H. (2013b). Beyond efficiency. *Commun. ACM*, 56(10):38–40. Author preprint: http://nm8.us/1.

- Ackley, D. H. and Cannon, D. C. (2011). Pursue robust indefinite scalability. In *Proc. HotOS XIII*, Napa Valley, California, USA. USENIX Association.
- Ackley, D. H., Cannon, D. C., and Williams, L. R. (2013). A movable architecture for robust spatial computing. *The Computer Journal*, 56(12):1450–1468.
- Bajec, I. L. and Heppner, F. H. (2009). Organized flight in birds. *Animal Behaviour*, 78(4):777 789.
- Bedau, M. A. (2003). Artificial life: organization, adaptation and complexity from the bottom up. *Trends in Cognitive Sciences*, 7(11):505 512.
- Bedau, M. A., McCaskill, J. S., Packard, N. H., and Rasmussen, S. (2010). Living technology: Exploiting life's principles in technology. *Artificial Life*, 16(1):89–97.
- Beer, R. D. (2014). The cognitive domain of a glider in the game of life. *Artificial Life*, 20(2):183–206.
- Cappello, F., Geist, A., Gropp, B., Kal, L. V., Kramer, B., and Snir, M. (2009). Toward exascale resilience. *IJHPCA*, 23(4):374–388.
- Dijkstra, E. W. (1974). Self-stabilizing systems in spite of distributed control. *Commun. ACM*, 17(11):643–644.
- Dittrich, P., Ziegler, J., and Banzhaf, W. (2001). Artificial chemistries: A review. *Artificial Life*, 7:225–275.
- Gardner, M. (1970). The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, 223:120–123.
- Grand, S. (2003). *Creation: Life and How to Make It.* Harvard University Press.
- Harding, L. (2014). *The Snowden Files: The Inside Story of the World's Most Wanted Man*. Vintage. Knopf Doubleday Publishing Group.
- Nakamura, K. (1974). Asynchronous cellular automata and their computational ability. *Systems, Computers, Controls*, 5(5):58–66.
- Nehaniv, C. L. (2004). Asynchronous automata networks can emulate any synchronous automata network. *IJAC*, 14(5-6):719–739.
- Perlroth, N. (2013). Target struck in the cat-and-mouse game of credit theft. *The New York Times*.
- Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 25–34, New York, NY, USA. ACM.
- Schneider, M. (1993). Self-stabilization. *ACM Comput. Surv.*, 25(1):45–67.
- von Neumann, J. (1951). The general and logical theory of automata. In Jeffress, L. A., editor, *Cerebral Mechanisms in Behaviour: the Hixon Symposium* (1948). Wiley.
- Xu, Y., Du, Y., Zhang, Y., and Yang, J. (2011). A composite and scalable cache coherence protocol for large scale CMPs. In Proceedings of the International Conference on Supercomputing, ICS '11, pages 285–294, New York, NY, USA. ACM.

Novel Approaches to the Visualization and Quantification of Biological Simulations by Emulating Experimental Techniques

James A. Butler¹, Kieran Alden¹, Henrique Veiga Fernandes², Jon Timmis¹, and Mark Coles¹

York Computational Immunology Lab, University of York, YO10 5DD, United Kingdom
 Instituto de Medicina Molecular, Faculdade de Medicina de Lisboa, 1649-028 Lisbon, Portugal jb1196@york.ac.uk

Abstract

The use of modeling and simulation as a predictive tool for research in biology is becoming increasingly popular. However, outputs from such simulations are often abstract and presented in a very different manner to equivalent data from the biological domain. Therefore, we have developed a flexible tool-chain for emulating various biological laboratory techniques to produce biologically homomorphic outputs in computer simulations. This includes virtual immunohistochemistry, microscopy, flow cytometry, and gene expression heatmaps. We present a case study in the use of this tool-chain applied to a simulation of pre-natal lymphoid organ development. We find that application of the tool-chain provides additional, biologically relevant data, that is inaccessible with pre-existing methodologies for analysis of simulation results. We argue that biological experimental techniques borrowed from the wet-lab are an important additional approach to the analysis of simulations in computational biology, and might furthermore inspire confidence in simulation results from the perspective of experimental biologists.

Introduction & Background

In silico simulations of biological processes, including disease pathology, tissue development, immune responses, and evolutionary processes, have a demonstrated ability to offer new insight into complex biological systems that are difficult to study solely in wet laboratories. Computational models offer qualitative and quantitative insights into complex systems, in which biological behaviors emerge from the interactions of many individual entities. For instance, by capturing the dynamics of cells signalling each other through both direct contact and through the secretion and detection of molecules in their local environment. We have previously demonstrated that agent-based spatially resolved simulations, when properly validated, have the capacity to explore disease intervention strategies such as the administration of drugs or biological therapeutics, or the simulation of surgery by removing or modifying individual model compartments (Read et al., 2013). Agent-based simulations of biology are often much more complex than mathematical models based on differential equations, as they capture emergent phenomena in terms of spatially-resolved individuals rather than at the population level. Such simulations often require a very large parameter space and fine spatial resolution. Therefore, it is extremely important that such simulations can be properly validated and calibrated in order for us to have confidence in their results and predictions.

Calibration is typically achieved in the first instance 'by hand', that is, the parameter space is explored in conjunction with a domain expert, until the simulation outputs correspond to those measured experimentally. Simulations are often calibrated against one experiment, which is unlikely to be adequate if the simulation is then used to explore the properties of the system in other contexts. For this reason, Read et al. (2013) argues for use of multiple calibration points when developing simulations. However, due to the limitations of observation in the biological domain, simulation outputs usually take on a very different form to in vivo or in vitro experiments that model the same process. Many simulations have parameters that are not directly measurable with current technology, which must be inferred from statistical analysis of the simulation results and calibration against primary data from biological experiments. The difficulty of this task may be compounded by the different nature of simulation output to primary data. Sensitivity analysis should be performed over the parameter space for every biological simulation, using techniques that measure the effect magnitude of each simulation parameter individually, such as the Vargha-Delaney A-Test (Vargha and Delaney, 2000), and Latin Hypercube parameter sampling to explore the effect different parameter values exert on the sensitivity of other parameters (Alden et al., 2013). Yet, we believe that in depth statistical analysis and parameter-fitting through calibration, does not provide enough evidence alone to convince a biologist that a simulation is fit for purpose, nor does it provide access to the model's full informational content.

To ensure that simulations are developed in a principled manner, we advocate use of the CoSMoS (Complex Systems Modeling and Simulation) framework for the development of computational models. We provide a brief description of the process below but direct the reader to Andrews et al. (2010) for a more complete overview. The model should be explicitly stated using a modeling language, such as the Uni-

fied Modeling Language (UML), Systems Biology Markup Language (SBML) or the Pi calculus, in terms of biology alone before conversion and abstraction into a platformindependent 'platform model', which may then be implemented through development of an executable software representation of the platform model. Simulation developers should also ensure the implementation is fully transparent: by providing a formal argumentation structure that explains and justifies all assumptions and abstractions with evidence or exposition, such that these can be considered when translating the simulation result into one grounded in the biological domain. Goal-structuring notation has been shown to provide a means of structuring such arguments in the field of safety-critical software systems (Kelly, 1999). Where simulations are used as a key tool in making biological predictions, clinical trials being one example, it is clear that the tool should be considered safety-critical.

Although these techniques may appeal to the developers of biological simulations, they are not always easily accessible to biologists, nor ideal for validation or predictive purposes. One asset of simulation is its capability of providing high-resolution data which is difficult or impossible to obtain in the wet-lab, while maintaing a level of abstraction that makes the simulation computationally tractable. However, it is for precisely these same reasons that experimental biologists often lack confidence in simulation results, so we argue that model developers need to go further to build confidence in their simulations. Simulation visualizations are commonly too abstract to visually represent a model system as it is conceptualized by biologists. Since simulation outputs do not reflect the format or type of data obtained from wet-laboratory experimental techniques, such as flow cytometry (to measure cell surface expression of specific proteins), histology, or the various approaches to analysing relative gene expression (quantitative polymerase chain reactions, microarrays, deep sequencing, etc), they must be indirectly compared with the biological domain.

In addition to providing transparency in simulation design, a strong argumentation structure, and a comprehensive statistical analysis of the parameter space, we argue modelers must look toward experimental techniques in biology with regard to the simulation outputs chosen. In this paper, we discuss the development of a tool-chain that enables simulations to output data comparable to biological experiments, through the emulation of experimental techniques used with *in vitro* or *in vivo* biological model systems. We additionally address the motivations and potential applications driving development of this approach.

The concept of producing a Turing-like test for the validation of biological simulations is discussed at length in Harel (2005), in which a domain expert is presented with both experimental and simulated datasets, and challenged to identify the experimental data, and whether any discernible difference exists. Much like the Turing test in the field of

artificial intelligence, this has been considered the standard to which computational simulations in biology should ultimately aspire to. The principal barrier to developing such a test as a viable validation tool, is negating the significant differences in means of producing, analyzing and presenting data in experimental and computational biology. Therefore, this presents a strong motivation to develop methods for bringing results in computational biology closer to those seen in experimental biology.

Essentially, we argue that to better understand the dynamics of a simulation of biological processes, an *in silico* emulation of experimental biological techniques is required, and furthermore, that a simulation is more likely to yield useful results when analyzed within the context of the biological techniques that will ultimately be used to test simulation predictions. We found that this process can elucidate new perspectives in a pre-existing model, and identify emergent sub-populations of cells not previously known within the simulation.

Table 1 presents the three wet-laboratory experimental techniques in biology that we aim to emulate in this case study: flow cytometry, histological imaging, and heat maps of protein expression both spatially and temporally expressed.

Technique	Description Illustrates	
Flow Cytometry	Cells are input via microflu- idics and individually scanned by lasers to determine the in- tensity of their fluorescent an- tibody stained surface.	Relative cell surface expres- sion of proteins, cell size and granularity. Can identify dif- ferent cell populations and is multi-dimensional.
(Immuno-)histology	Sections of tissue are antibody stained and imaged using mi- croscopy.	Detect the presence of pro- teins and tissue structures. Spatially resolved.
Heat Maps	Used to illustrate gene expres- sion data, typically from mi- croarrays or deep sequencing.	visualizations of spatiotempo- ral gene or protein expression (relative).

Table 1: Table of experimental techniques in biology to be emulated and applied to artificial life simulations.

Methodology

A typical approach to experimentation with an existing computational biology simulation is to perform various statistical analyses on simulation outputs for a range of parameter samples. These are then used in an attempt to make predictions about a biological system, which leads to further experimentation with the simulation, and also predictions that could be tested in the web laboratory. However, there does not exist a principled approach to linking the results model of a simulation back to the original biological domain model.

We propose the creation of a stronger link between biological models and executable simulations in general by developing simulations that produce outputs that map directly to the types of data produced and used by experimental biologists. It is to this end that we have developed software and

protocols for the production of such datatypes that can be applied to pre-existing and new simulations, and enable computational models to be better interpreted within the context of the biology they represent. Figure 1 presents our standard workflow (black rectangles) and incorporates the proposed additional steps (red rectangles) for creating models and simulations that better integrate with experimental biology. The net result of these transformed output is a wider range of outputs that can be used during simulation calibration and validation, and to aid development of directly testable hypotheses in terms of wet laboratory experimentation.

Through the adoption of an agent-based modeling approach, we previously created a computational predictive tool for exploring the mechanisms driving pre-natal lymphoid organ development. This has aided the generation of testable biological hypotheses concerning the complex cellular interactions leading to the generation of organs that trigger adaptive immune responses: interactions which cannot currently be fully explored using laboratory techniques. For the purpose of this paper, we give a brief overview of the model, but direct the reader to our previously published work detailing the simulation design, implementation and analysis (Alden et al., 2012; Patel et al., 2012; Alden et al., 2013). The tool captures the 72 hour period of tissue development in pre-natal mice. Populations of hematopoietic cells, known as Lymphoid Tissue Initiator (LTin) and Lymphoid Tissue Inducer (LTi) cells, migrate into the developing gut, with data from laboratory observations suggesting these cells follow a random motion. Both cell populations express receptors for the adhesion molecule VCAM-1, expressed by stromal Lymphoid Tissue Organizer (LTo) cells residing in the gut wall. VCAM-1 causes cells expressing the cognate receptor to adhere to the VCAM-1 expressing cell, thereby restricting its movement. Contact between a hematopoietic cell and LTo cell triggers the LTo cell to differentiate (become more specialized), leading to increased adhesion molecule expression. In addition, LTo cell differentiation increases chemokine secretion, creating a chemokine gradient that promotes migration of the LTi cell population towards the differentiated LTo cell. In the vicinity of LTo cells, movement of LTin and LTi cells will be restricted by adhesion factors (VCAM-1 and others), forming aggregations of hematopoietic cells around LTo cells by the end of the 72 hour period. These aggregations later mature into lymphoid organs called Peyer's patches, which are capable of initiating immune responses against pathogenic bacteria encountered in the gut.

We have previously shown that the emergent cell behavior observed in laboratory experimentation is statistically similar to that observed in the simulation. We utilized sensitivity analysis techniques to explore the simulated biological pathways to reveal those which have a significant impact on simulation response (Patel et al., 2012; Alden et al., 2013). The

output from these statistical techniques provided evidence that our simulation is fit for the purpose of aiding biologists in their exploration of the system. However, we believe that confidence in the simulation would be further increased by providing experimental biologists with simulator outputs that are comparable to primary laboratory data, which can be more intuitively interpreted. Furthermore, such outputs can provide additional insight into the simulation dynamics, enable additional explorative experimentation *in silico* and furnish mechanistic detail not readily accessible with descriptive nor inferential statistical analyses.

In order to emulate flow cytometry, a simulation requires explicit values for cell surface expression of proteins. Absolute values are not important, rather, it is the relative differences in expression levels that enables the insight afforded by this technique. In the Peyer's patch simulation, expression is strictly binary, such that each cell in the simulation is either expressing a protein or not. An increase in protein expression is achieved by changing the parameters of abstract mathematical functions that determine behavior. For example, LTo chemokine expression levels are abstracted as a sigmoidal cumulative probability density function, sampled at each time step by every LTi cell responsive to chemokine. An increase in chemokine expression level is represented as a reduction in the standard deviation (or tightness) of the sigmoid curve, as detailed in Alden et al. (2012). Expression of VCAM-1 by LTo cells is handled in a similar manner. Therefore, it is necessary to modify the simulation to provide relative quantitative expression levels without compromising extant simulation dynamics. In these cases, a new agent property needs to be created that represents relative expression of the factor concerned, but is not used by the simulation for decision making purposes. This value is obtained by incrementing (or decreasing) the property each time step in direct proportion to the change in the mathematical function responsible for the protein.

For each time-point a flow cytometric analysis is to be performed, expression levels of each protein of interest are placed into a CSV column and multiplied by a factor of 10^5 prior to bring cast as integers. This ultimately enables flow cytometry software to interpret the values as fluorescent intensities, which are proportional to expression level. Following this, the CSV data is transformed into 'fcs' format compliant files according to the standard described in Spidlen et al. (2010), which are universally used by modern flow cytometers to store their data output. This is an important step because it allows the data to be interrogated using flow cytometry software, specifically designed for use with biological data.

For this case study, we chose to emulate flow cytometry for the generation of adhesion molecule VCAM-1 expression data. VCAM-1, expressed by LTo cells, is responsible for retaining LTi cells in Peyer's patch formation. We also emulate the forward scatter (FSC) of light that occurs when

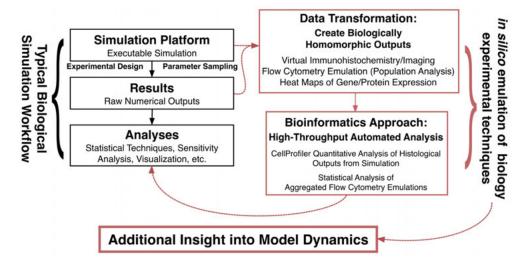


Figure 1: Our typical approach to performing and analyzing simulations in computational biology. The dashed lines represent additional work-flow designed to provide outputs homomorphic to experimental data, which enables additional analysis of model dynamics and direct comparison with experimental data from the wet lab.

lasers strike the cell in the flow cytometer, which is proportional to the size of the cell. In the simulation, cells of a given phenotype have the average diameter of those cells as measured experimentally. We adapted the simulation such that the radius is sampled from a truncated gaussian distribution about the mean experimentally measured radius, with a standard deviation of $1\mu m$. This enables identification of cell populations based on both their size and their expression of VCAM-1.

The simulation is written in Java, and utilizes MASON (Luke et al., 2005), a domain-independent agent-based modeling toolkit. The simulation visualizations in this paper are not MASON dependent, and are also written in Java. Perl and Octave are used to manipulate output CSV data and generate 'fcs' compliant files for use in flow cytometry data analysis software.

Results

This section presents results from the adapted simulation, transformed into biologically homomorphic outputs as an emulation of histology and microscopy, flow cytometry, and relative protein expression. We also explore the application of software designed to automate the analysis of biological images, enabling a high-throughput approach to analysing graphical outputs representing histological staining and microscopy.

Figure 2 presents new visualization approaches derived from the existing simulation of Peyer's patch formation. These represent 'unfolded' images of a section of mouse gut, as the simulation is toroidal about the Y axis. All cells in the simulation are stored in a 2-dimensional continuous grid object provided by the MASON agent-based simula-

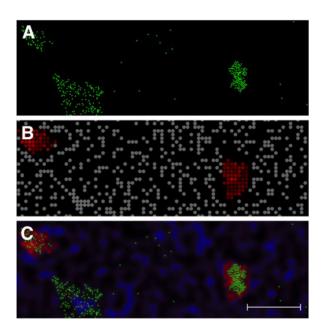


Figure 2: Simulation visualization after 72 hours. **A.** LTi cells colored green to simulate GFP in microscopy. **B.** LTo cells in grey, and level of red proportional to VCAM-1 expression level. **C.** Complete emulated histology and microscopy image showing two Peyer's patches. LTi cells are stained green, LTo cells stained blue, with additional VCAM-1 staining in red. Scale bar 175 microns.

tion toolkit (Luke et al., 2005). Each object of a particular cell type is extracted from this grid and then drawn to a canvas according to the properties of that cell, as seen in typical

immunohistological imaging. In Figure 2A, LTi cells are drawn as green circles ((0,255,0) in the RGB colorspace) on a black canvas, in a manner similar to the appearance of GFP (green fluorescent protein)-stained cells when imaged with confocal microscopy. Overlapping cells produce a region of green with a higher *alpha* (transparency) value, to enable determination of the density of a region by measuring the level of alpha in that region as compared with the base level assigned to individual cells. LTo cells are illustrated in Figure 2B, drawn red (255,0,0), with an alpha value corresponding to VCAM-1 expression level. LTo cells without sufficient VCAM-1 expression have been colored grey (128,128,128) for the purposes of the figure to be rendered visible to the reader.

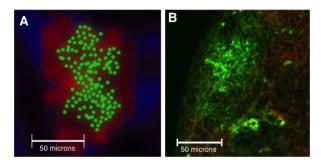


Figure 3: **A.** Emulated immunohistology and microscopy of a Peyer's Patch at Embryonic Day 17.5, simulated *in silico*. LTi cells are green, the level of red is directly proportional to VCAM-1 expression on LTo cells, while blue indicates the presence of LTo cells not expressing significant levels of VCAM-1. **B.** Actual confocal microscopy image of an antibody stained B Cell follicle within a developing murine lymph node. The B Cells are stained green, the surrounding T cells are stained red. Scale bars are both 50 microns.

The two canvases undergo several stages of post-processing before being combined into Figure 2C, as an emulation of immunohistology that shows the co-localization of VCAM-1 expression on LTo cells, and LTi cells. The VCAM-1 canvas undergoes posterisation and a gaussian blur, to provide an interpolated, continuous approximation of VCAM-1 expression in Peyer's patches. Furthermore, a canvas with dark blue circles representing all LTo cells undergoes a gaussian blur and is drawn at 25% transparency, beneath the red VCAM-1 layer. The LTi cell layer is duplicated, with the lower layer undergoing blurring and an increase in transparency to 25% visibility.

An enlarged section of Figure 2C is shown in Figure 3A, illustrating an individual Peyer's patch *in silico*, alongside an actual confocal microscopy image in Figure 3B of a B Cell follicle in a developing lymph node, included to illustrate how genuine lymphoid tissue histology compares to the *in silico* emulation. The development process of Peyer's patches and lymph nodes are qualitatively similar, however

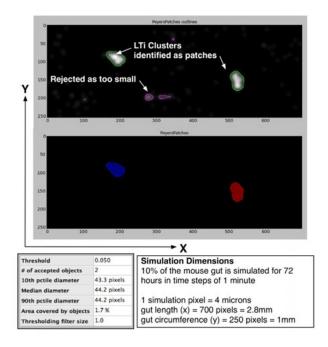


Figure 4: CellProfiler output for one simulation run in which two Peyer's Patches are identified, and one cluster of LTi cells is discarded due to insufficient area. This analysis can be automated over many thousands of runs to identify trends in patch size in terms of both area, density and compactness.

it is important to note that the cell types and staining in Figure 3B are not the same as those emulated in Figure 3A.

The *in silico* images are output from the simulation at any pre-specified resolution, with a minimum usable value depending on the diameter of the smallest visualized object. The images may be automatically analyzed en masse to detect the presence of Peyer's patches using CellProfiler pipeline-based image analysis software (Carpenter et al., 2006) for quantitatively identifying and analysing cell phenotypes. This is illustrated in Figure 4 for a single simulation run, but can be applied to an arbitrarily large dataset, using computer clusters if necessary. Prior to the development of this CellProfiler pipeline, identification and quantification of patch formation has proven difficult, as it is achieved in the domain through manual analysis of histology, by eye (Cornes, 1965). The same CellProfiler pipeline has also successfully been applied to the measurement of developing lymphoid tissue histology obtained using confocal microscopy (data not shown).

The raw fcs formatted data can be analysed in a variety of ways using flow cytometry software. For this case study, we used WEASEL v3.1 (Walter and Eliza Hall Institute of Medical Research). In Figure 5A, a dot-plot of VCAM-1 vs Forward Scatter (FSC) florescent intensity at 72 hours is shown, for every cell in the simulation. This enables identification

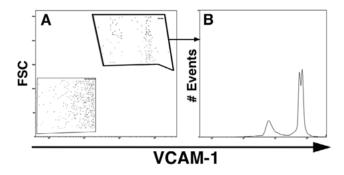


Figure 5: **A.** Emulated FACS plot of VCAM-1 Mean Florescent Intensity (MFI) vs Forward Scatter (FSC). This is then gated on the VCAM-1 positive cells with high forward scatter, which are the large stromal cells. From this subset, **B.** shows a histogram of VCAM-1 MFI vs. the number of recorded events (cells). These plots were generated using WEASEL Flow cytometry data analysis and display software (Walter and Eliza Hall Institute of Medical Research), intended for use with biological data from FACS machines.

of two main populations of cells present at the 72-hour timepoint: a larger VCAM-1 high, high FSC population and a smaller, VCAM-1 negative, low FSC population. Closer examination reveals two populations of VCAM-1 positive large cells, separated by an order of magnitude in VCAM-1 expression. In Figure 5B, the VCAM-1 positive, large cells are isolated and presented as a histogram of VCAM-1 mean florescent intensity. Two peaks in population can be observed, corresponding to the two sub-populations identified in Figure 5A.

The flow cytometry emulation data can be output from the simulation at any desired time-points. This enables flow cytometry to be performed repeatedly at different time-points from the same simulation run, and permits observation of the progression of cell populations over time. To illustrate this, the histogram from Figure 5B is shown at 24-hour intervals over the 72-hour formation process. The time-dependent emergence of the peak population size and VCAM-1 expression level is clearly visible.

Mean protein expression levels can be output from the simulation both over time and across space. This permits the creation of heat maps that capture cell phenotypes spatiotemporally, and can show the progression of protein expression over time and space. Figure 7A shows a sample of 9 cells and their expression of VCAM-1 and chemokine CCL19 at three different time-points. This illustrates the activity through time of particular proteins or genes; in this example, both VCAM-1 and the chemokine CCL19 show significantly more expression during the final 24 hour time-point. In Figure 7B, the spatially-distributed expression of VCAM-1 is shown at 72 hours for three isolated Peyer's patches that formed during one simulation execution.

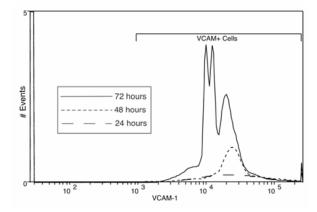


Figure 6: VCAM-1 expression histogram for all cells, taken at 24 (long dash), 48 (short dash), and 72 (solid line) hour time-points. The emergence of a prominent peak is visible after 48 hours, and by the end time-point this has diverged into several distinctly identifiable sub-populations, each corresponding to VCAM-1 expression levels within a specific Peyer's patch.

Discussion

The development of simulation outputs that directly reflect data obtained from the laboratory confers several advantages when used to augment current approaches to simulation analysis. Sensitivity analysis and other descriptive statistical methods are extremely important in evaluating simulations, and have proven useful in determining simulation robustness, aleatory uncertainty, and the roles of specific parameters with respect to particular outputs (Marino et al., 2008; Read et al., 2012; Alden et al., 2013). However, they do not offer the mechanistic insight into the spatial organisation of cells and the structures they form, or the changing cell phenotypes and emergent populations that may be observed using multiple time-point emulated FACS analysis, emulated histology and spatially resolved heat-maps. We therefore argue that current best-practice simulation analysis methodologies should be augmented with emulations of biological experimental techniques. Sensitivity analysis may then be performed that determines the effect magnitude of parameters on more biologically relevant outputs.

The application of the histology emulation module and CellProfiler pipeline to the Peyer's patch simulation has created potential for a new set of experiments that not previously possible. These are to explore how the number of patches, and their morphology, change in response to parameter perturbation and gene-knockouts. Previously, experimentation focused on velocity and displacement of LTi cells as the primary outputs of interest, however we can now measure a large range of properties relating to both the structure of the patches such as compactness, density and area, and also the spatial organization of expressed proteins. Previ-

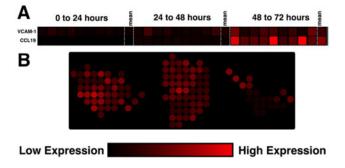


Figure 7: Heat-maps generated from simulation data over both time and space, to illustrate change in protein expression levels. **A.** VCAM-1 and CCL19 expression over time for a subset of individual LTo Cells and then the mean level over that subset. **B.** Spatially-resolved heat-map showing VCAM-1 distribution in three Peyer's Patches that formed during one simulation run.

ously, experiments relating to patch number were intractable due to the large number of replicates per parameter sample required to ameliorate aleatory uncertainty and the lack of a suitable means of identifying and counting patches; clustering algorithms were used in an attempt to count patches based purely on the locations of LTi cells, however they were found to be unreliable. New insight into the model dynamics arose from the simple emulated flow cytometry analysis performed in Figures 5 and 6. There is a clear emergence of two distinct populations of LTo cells in Figure 5A. Interestingly, we learn from Figure 6 that there is a sudden divergence after 48 hours that created these populations, before which expression levels conformed to a gaussian distribution - an emergent effect in itself. The divergence could be a simulation artifact or it could be that each patch has a relatively uniform distribution of VCAM-1 dependent on its unique properties, such as size and population size of co-localized LTi cells. However, any prediction made from the emulated experimental technique is a prediction arising from the simulation itself, in that the techniques described herein should not effect extant simulation behavior. Such predictions were merely unrecoverable with previously applied analytic techniques. Therefore, validation of the these techniques must be considered within the context of the simulation to which they are applied, as what is a suitable approach for one simulation may be wholly inappropriate when applied to another.

It has been suggested several times throughout development of the field of computational biology, that a reverse-Turing test (Sargent, 1991; Harel, 2005), in which both biological data and simulated data are presented to a biologist, in order to observe whether the simulated dataset can be readily identified, and how. This has largely not been feasible to-date because of the very different manner in which simulation results are presented, and the difficulty in buffer-

ing these differences from biological domain experts. Now, with the availability of simulated histology and flow cytometry data, we are quickly approaching the point at which this will become feasible. For instance, prior to the development of the CellProfiler pipeline illustrated in Figure 4, there was no reliable, automated means by which to identify, count and quantify Peyer's patches over many simulation run results, and simple dots were used for each cell in the simulation visualization layer. Furthermore, there are no other means except emulated flow cytometry to perform an in depth analysis of cell populations and their properties within the simulation. The possibility of simulation and experimental model results becoming indistinguishable is an exciting prospect in terms of accelerating scientific progress in biology and medicine.

Experimental biologists experienced with flow cytometry could identify important patterns, populations and other results that could go unnoticed in other approaches to simulation analysis. Flow cytometry software such as FlowJo (TreeStar) and WEASEL (Walter and Eliza Hall Institute of Medical Research) enable gating of events based on the value of multiple parameters, permitting identification of different phenotypes, and sub-populations within those phenotypes, and an exploration of the properties of these populations. Although for the sake of simplicity and demonstration purposes we chose to emulate flow cytometry with just two factors, depending on the complexity of the simulation, this approach can be extended *in silico* to an arbitrarily large number of expressed proteins. The multi-dimensional nature of flow cytometry data coupled with software designed specifically for its analysis in a biological context is what makes this approach particularly appealing. Attempting to perform the same sort of analyses on cell populations and phenotypes using the simulation alone, without flow cytometry emulation, would be extremely time consuming and difficult.

The temporally- and spatially- resolved heat-maps presented in Figure 7 have the potential to be combined in various novel ways to demonstrate the development of biological structures through changing gene expression across time and space. In fact, this is an example of a technique grounded in experimental biology, but developed beyond the limits of what is possible experimentally. The same may be said of the flow cytometry analysis performed at multiple time-points: in experimental biology, the act of running a sample through a FACS machine irretrievably destroys it, rendering evaluation of an identical sample at earlier or later time-points impossible in the wet-lab. Therefore, while providing data in a format familiar to biologists, these techniques do not limit simulation results to what is possible within the confines of a wet-lab. We argue that the approaches developed in experimental biology for exploring and visualizing data, did so because of their usefulness in interpreting biological systems, and that these approaches should be broadly extended also to *in silico* models of biological systems.

Through the application of high performance, highthroughput computing, emulated experimental technique results from many simulations can be analyzed and combined to provide a realistic quantitative analysis of biological simulations to provide data that can be more easily integrated into biological experiments, and enable direct comparison between computational and wet-lab datasets. This approach has the potential to somewhat simplify simulation calibration, given the data output from the simulation takes the same form as biological data, the fitness function of the simulation effectively becomes the primary biological data, allowing systematic exploration of the parameter space to identify the points at which the simulation output is statistically no different from domain experimental data. Combined with genetic algorithms or other evolutionary computation approaches, the speed and accuracy by which simulations may be automatically calibrated by computer could be improved significantly.

In summary, we have presented a methodology and toolchain for generating simulation outputs that are homomorphic to primary biological data. We argue that this improves the validation process, and by allowing direct comparison with the domain, it enhances the confidence one may have in a biological simulation. The tool-chain developed has enabled novel quantification and visualization of computer simulations that was hitherto unachieved, and allows exploration of model dynamics that would have otherwise remained hidden when limited to statistical analyses. We have argued that biological simulations should aim to produce the same data format as that which was used to create it.

Future Work

The groundwork has been laid for a broadly applicable toolkit that may be used to generate histological, FACS and gene expression data from computational simulations. We plan to build upon the histological aspect of the tool-chain by creating a library of drawing and image processing techniques to replicate a wider range of cell phenotypes. The long-term goal is a freely distributable software tool that can be integrated with a wide range of simulations, to produce more biologically relevant outputs that can both inspire confidence that simulations are fit-for-purpose, and improve the simulation analysis process.

Acknowledgements

This work is supported by the Engineering and Physical Sciences Research Council, United Kingdom.

References

Alden, K., Read, M., Timmis, J., Andrews, P. S., Veiga-Fernandes, H., and Coles, M. (2013). Spartan: A comprehensive tool for understanding uncertainty in simulations of biological systems. *PLoS computational biology*, 9(2):e1002916.

- Alden, K., Timmis, J., Andrews, P. S., Veiga-Fernandes, H., and Coles, M. C. (2012). Pairing experimentation and computational modeling to understand the role of tissue inducer cells in the development of lymphoid organs. *Frontiers in immunology*, 3.
- Andrews, P. S., Polack, F. A., Sampson, A. T., Stepney, S., and Timmis, J. (2010). The cosmos process version 0.1: A process for the modelling and simulation of complex systems. *Department of Computer Science, University of York, Tech.* Rep. YCS-2010-453.
- Carpenter, A. E., Jones, T. R., Lamprecht, M. R., Clarke, C., Kang, I. H., Friman, O., Guertin, D. A., Chang, J. H., Lindquist, R. A., Moffat, J., et al. (2006). Cellprofiler: image analysis software for identifying and quantifying cell phenotypes. *Genome biology*, 7(10):R100.
- Cornes, J. (1965). Number, size, and distribution of peyer's patches in the human small intestine part i the development of peyer's patches. *Gut*, 6(3):225–229.
- Harel, D. (2005). A turing-like test for biological modeling. *Nature Biotechnology*, 23(4):495–496.
- Kelly, T. P. (1999). Arguing safety: a systematic approach to managing safety cases. University of York.
- Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., and Balan, G. (2005). Mason: A multiagent simulation environment. Simulation, 81(7):517–527.
- Marino, S., Hogue, I. B., Ray, C. J., and Kirschner, D. E. (2008). A methodology for performing global uncertainty and sensitivity analysis in systems biology. *Journal of theoretical biology*, 254(1):178–196.
- Patel, A., Harker, N., Moreira-Santos, L., Ferreria, M., Alden, K., Timmis, J., Foster, K., Garefalaki, A., Panayotis, P., Andrews, P., Enomoto, H., Milbrandt, J., Pachnis, V., Coles, M. C., Kioussis, D., and Veiga-Fernandes, H. (2012). Differential ret signalling pathways drive development of the enteric lymphoid and nervous systems. *Science signaling*, 5(235):ra55.
- Read, M., Andrews, P. S., Timmis, J., and Kumar, V. (2012). Techniques for grounding agent-based simulations in the real domain: a case study in experimental autoimmune encephalomyelitis. *Mathematical and Computer Modelling of Dynamical Systems*, 18(1):67–86.
- Read, M., Andrews, P. S., Timmis, J., Williams, R. A., Greaves, R. B., Sheng, H., Coles, M., and Kumar, V. (2013). Determining disease intervention strategies using spatially resolved simulations. *PloS one*, 8(11):e80506.
- Sargent, R. (1991). Simulation model verification and validation. In *Proceedings of the 1991 Winter simulation Conference*, pages 37–47. IEEE Computer Society.
- Spidlen, J., Moore, W., Parks, D., Goldberg, M., Bray, C., Bierre, P., Gorombey, P., Hyun, B., Hubbard, M., Lange, S., et al. (2010). Data file standard for flow cytometry, version fcs 3.1. *Cytometry Part A*, 77(1):97–100.
- Vargha, A. and Delaney, H. D. (2000). A critique and improvement of the cl common language effect size statistics of mcgraw and wong. *Journal of Educational and Behavioral Statistics*, 25(2):101–132.

Easing Parameter Sensitivity Analysis of Netlogo Simulations using SPARTAN

Kieran Alden^{1,2,3}, Jon Timmis^{1,3} and Mark Coles^{1,2}

¹York Computational Immunology Laboratory, University of York. UK
²Centre for Immunology & Infection, Hull York Medical School and Dept of Biology, University of York. UK
³Department of Electronics, University of York. UK
kieran.alden@york.ac.uk

Abstract

In attempts to further understand complex systems at an individual level, the application of agent-based modeling is becoming prevalent across a range of academic disciplines. With the advantages of being multi-platform, requiring little programming experience, and supported by a large number of freely available case study examples, Netlogo has become a popular choice as the software tool to apply in the construction of agent-based models. To utilize the constructed model as an informative or predictive tool, statistical analyses can be performed to reveal the influence that a parameter has on simulation behavior, offering an insight into the system under study. Here we demonstrate the integration of Netlogo's parameter sweep function, Behavior Space, with an extended version of SPARTAN, our previously published open source statistical package for performing local and global sensitivity analyses. With the addition of SPARTAN, the researcher can automatically create Netlogo experiment files for both local (individual parameter) and global (latin-hypercube and Fourier frequency) analyses, run these experiments in Netlogo, and receive detailed statistical information on the influence a parameter has on simulation response: vital information for translating a simulation result to a hypothesis grounded in the system being studied. To ensure our example work is reproducible, we demonstrate use of SPARTAN using the Virus transmission and perpetuation model available in the Netlogo model library.

Introduction

As the application of agent-based modeling has become adopted across a range of academic disciplines, including biological, sociological, and economic, much focus has been placed on the release of software tools to aid the generation of a simulation of a system under study. One such popular toolkit is Netlogo (Wilensky, 1999), a freely available, multi-platform toolkit that aims to ease the construction of multi-agent systems. Simulations have been generated for a wide range of applications, recent examples being to inform government energy policy (Lee et al., 2014), to simulate trout demographics in a brook system (Frank and Baret, 2014), and emergency procedure planning for building evacuation (Wagner and Agrawal, 2014).

Previously we have demonstrated that our bespoke agentbased model, not generated in Netlogo, can create emergent cell behavior that is statistically similar to that observed experimentally (Patel et al., 2012; Alden et al., 2012), providing a tool through which hypotheses concerning the formation of lymphoid tissue formation can be developed and tested. In the course of understanding the relationship between our simulation and the biological system that it represents, we developed SPARTAN (Simulation Parameter Analysis R Toolkit ApplicatioN) (Alden et al., 2013), a package of statistical techniques specifically designed to aid researchers in translating a simulation result to an hypothesis grounded in the real-world domain. SPARTAN is open source and freely available from either the R package repository (CRAN) or from www.ycil.org.uk/software/spartan. To the best of our knowledge this was the first time a comprehensive package had been made available that could be applied to agent-based as well as traditional ordinary and partial differential equation models. Alongside the release of the package, we demonstrated the potential the included parameter sensitivity analysis techniques have in revealing the influence particular simulated pathways and components could have on the biological system.

The initial demonstration of SPARTAN utilized our lymphoid tissue simulator, a bespoke agent-based software tool created in Java with the aid of the MASON Simulation toolkit (Luke, 2005). In this case, we utilized SPARTAN in the creation of parameter value sets, performed simulation runs under the generated parameter conditions, and used the relevant SPARTAN technique to analyze the results. With this approach proving fruitful in understanding the behavior of our simulator, we have recently extended SPARTAN to enable researchers who use Netlogo to perform both local and global sensitivity analyses of their agent-based models. Although Netlogo is equipped with the Behavior Space feature, which allows the researcher to perform a sweep of potential parameter values, the researcher is required to develop routines for analyzing the resultant simulation responses. Recent work has been undertaken to link Netlogo with statistical environments such as R (Thiele et al., 2012), but does not provide statistical algorithms to fully understand the relationship between a simulation and the system it represents. Here we demonstrate how combining Netlogo with SPARTAN provides the researcher with the tools to perform both parameter sampling and the analysis of the resultant simulation data, for three sensitivity analysis techniques. Firstly we describe the model that we have chosen to use, a virus transmission and perpetuation model (Wilensky, 1998) that is available in the Netlogo Model Library. We then briefly discuss the three sensitivity analysis techniques that are included within the SPARTAN package (one local analysis, two global analyses), before describing the result of each analysis for simulation responses generated from the Netlogo model. This paper is supported by a full tutorial demonstrating how each technique can be applied to the virus transmission model, available both within the SPARTAN package and accompanying website (www.ycil.org.uk/software/spartan).

The Case Study: Virus Transmission and Perpetuation

To aid the adoption of SPARTAN by researchers utilizing Netlogo, we believed that it was important the demonstration we include here and in the tutorial can be easily replicated. Thus we utilize a simulator that is freely available in the Netlogo Model Library, with a number of modifications that we describe here.

The Virus model (Wilensky, 1998) simulates a human population, to suggest how a virus transmits and is perpetuated amongst that population based on factors suggested by ecological biologists (Yorke et al., 1979). A full description of the composition of the model is included with the simulation. For our interests, there are four parameters that can be set by the researcher utilizing the model: the number of people in the population (*people*), the ease at which the virus spreads (*infectiousness*), the probability a person recovers from the virus (*chance-of-recover*), and the duration in weeks after which the person either recovers or dies (*duration*). Here we will demonstrate the use of SPARTAN in determining how both robust these parameters are to perturbation and the relative influence of each parameter on simulation response.

Such judgments are dependent on an analysis of how the chosen parameter values impact a set of simulation responses at a specified simulation time-point. As the current version of the virus model has no defined end point, we have modified the simulator such that the simulation runs for a 100 year period. With the simulator specifying that a persons lifespan is 27 years if they are not killed by the virus, modeling an 100-year time-period is sufficient for our needs. In terms of output response, the simulator provides the percentages of the population that are infected and immune. As we want our analysis to be representative of the entire period rather than a snapshot of the population, we introduced four additional output measures: the number of people who have died through not recovering from the in-

fection (death-thru-sickness), the number who died but were immune (death-but-immune), the number who died through old age and never caught the infection (death-old-age), and the number of people who died while infected but during the time period allowed for recovery (death-old-and-sick). As we are utilizing statistical tests to determine how alterations to the four input parameters described in the previous paragraph affect these responses, which are all numbers of people, we exclude the people parameter from the analysis, and run all our experiments with an initial population of 150 people, leaving three input parameters of interest. Additionally, to run the eFAST analysis that will be described later, an additional global parameter is added ('dummy'), that has no impact on simulation response. The reasoning for this becomes clear in later sections of this paper.

To ease the reproduction of results in the latter sections of this paper, we have made the modified version of the model available from the SPARTAN website (www.ycil.org.uk/software/spartan)

Overview of Sensitivity Analysis Techniques in SPARTAN

In this section, we note the three parameter sensitivity analysis techniques available within the SPARTAN package. As we have previously described the full detail of the implementation of each technique, we cover each only briefly, and would direct you to our previously published work for the complete description (Alden et al., 2013; Read et al., 2012). The section that follows this details the application of each technique to the Netlogo Virus model.

Technique 1: Parameter Robustness

The parameter robustness technique examines the implications of any uncertainty or parameter estimation on simulator response. This highlights any parameters for which the simulation is sensitive to value perturbation, potentially suggesting that caution should be applied as a response may be an artifact of parameterization and not representative of the system that has been captured (Helton, 2008).

Parameter robustness is determined by perturbing each parameter independently of all others, which remain at their baseline value. Simulation responses under the perturbed conditions are compared with simulation response for baseline parameter values using the Vargha-Delaney A-Test (Vargha and Delaney, 2000), an effect-magnitude test used to determine if there is a statistically significant difference between simulation responses under differing conditions. For each parameter, SPARTAN produces two plots: the first detailing the A-Test response for each value it has been assigned, easing identification of parameter values that may cause a significant change in behavior, and a second showing the distribution of simulation responses for each parameter value.

Technique 2: Global Sensitivity Analysis using Latin-Hypercube Sampling

Whereas robustness analysis perturbs each parameter individually, both this and the next technique seek to identify any compound effects that are revealed when the values of all parameters of interest are perturbed simultaneously. This global sensitivity analysis exposes cases where the effect one parameter has on the simulation response is dependent on the value of another, and can indicate the parameters having greatest influence on simulation response. Where a simulation bears a strong relationship to the real system being modeled, such an analysis has the potential to offer significant insight into the system under study.

This technique selects a number of simulation parameter value sets using latin-hypercube (LHC) sampling. LHC selects a value for each parameter from a specified range, while ensuring efficient coverage of the parameter space and minimizing correlations between chosen value sets (Read et al., 2012; Marino et al., 2008; Saltelli et al., 2000). Simulations are then performed for each generated set of parameter values. A plot is produced by SPARTAN for each parameter-simulation response pairing, revealing correlations between parameter value and simulation response. For reporting purposes, this correlation is also quantified through calculation of the Partial Rank Correlation Coefficient (PRCC), which is reported in the plot header.

Technique 3: Global Sensitivity Analysis using Extended Fourier Amplitude Sampling Test

Whereas Technique 2 identifies compound effects using a sampling-based method (the hypercube), the extended Fourier amplitude sampling test (eFAST) (Saltelli, 2004; Saltelli and Bollardo, 1998) has been included in SPARTAN to offer an alternative: global analysis by variance decomposition. Variance in simulation output is partitioned between the parameters of interest, providing a statistical measure that reveals the proportion of variance that can be explained by the perturbation of each parameter. Such a measure is a key indicator that a parameter has a highly influential effect on simulation response.

Of the three techniques we are describing here, this is by far the most complex, and the reader is directed to our previously published work for the full detail of the algorithm (Alden et al., 2013). As a brief overview, each parameter is taken in turn as that of interest. These are accompanied by an extra parameter, the 'dummy', which has an arbitrary range of values and no impact on simulation response. Sinusoidal curves of a particular frequency are created in the parameter space, with the parameter of interest assigned a frequency that is significantly different to the remaining parameters in the analysis. From each curve, a specified number of values are chosen, in turn creating a number of parameter value sets for that parameter of interest (see Marino et al., 2008 for an illustration of this approach). A phase shift is employed that

shifts the frequency slightly, and the sampling repeated to mitigate the potential for identical value sets to be selected. This leads to the selection of a number of parameter value sets for each parameter, for each phase shift.

Simulations are executed for each of the parameter value sets generated in this process. The responses are analyzed by SPARTAN, taking into account the frequencies that were used in the generation of the parameter set. Fourier analysis is used to partition variance in simulation response between the parameters, and two statistical measures produced for each parameter: the fraction of output variance that can be explained by that parameter (Si) and the output variance caused by higher-order non-linear affects between the parameter and the others being studied (STi). A parameter is judged to have a significant impact on simulation response through contrasting these two measures with those calculated for the 'dummy' parameter that is known to have no impact.

Application of SPARTAN to Netlogo Model

This section demonstrates the application of the techniques above to the Netlogo Virus model described previously. In these analyses we treat the model default parameter values are those that create the simulation baseline behavior (chance-of-recover=50%, infectiousness=60%, duration=20 weeks), and assign a range of values over which the parameter will be explored (chance-of-recover=10-90%, infectiousness=10-90%, duration=5-40 weeks). For technique 1, we set an amount by which the parameter value will be incremented (1%, 1% and 5 weeks respectively). Each simulation was run using the Headless mode of Netlogo, allowing simulations to be run in batches from the command line. For each run, the simulation state is saved at each timestep using the table output functionality included in Netlogo's BehaviorSpace tool. We stress that no additional output function has been added to the model for our purposes, this is produced by Netlogo itself. This ensures the techniques demonstrated here are applicable to the output of any Netlogo model.

Technique 1: Parameter Robustness

Using the specified parameter space values, SPARTAN generates a Netlogo BehaviourSpace experiment XML file. Netlogo uses this file to perform a sweep of all potential values for that parameter. When complete, SPARTAN can process the simulation response file, taking each parameter in turn and performing a statistical comparison of simulation response when the parameter value is perturbed against behavior at baseline parameter values.

The analysis produces two forms of graphical output, both demonstrated in Figure 1. Figure 1(A) indicates how statistically different each simulation response is for each value assigned to a parameter, in this case the ease at which the virus

spread (*infectiousness*). For there to be no difference, the A-Test result would be 0.5. Large differences occur either side of this, although the direction is not of concern, just the indication of effect. In the case of Figure 1(A), the analysis reveals that a statistically significant alteration is observed for each simulation response with a perturbation of just 1%. Such a result indicates that the simulation is very sensitive to the value of this parameter, and this needs to be assigned carefully.

Figure 1(B) and 1(C) are demonstrations of the result distribution that is also generated for each simulation output measure, for each parameter being examined. In the case of Figure 1(B), this shows how an alteration in the probability of recovery impacts the number of people who die through infection by the virus. This shows the interesting effect that as the chance of recovery initially increases, the number of people who die from the virus also initially increases, up to a point where the number begins to fall. This may not be expected behavior, and may need to be investigated further to determine if there are links between this parameter and the value assigned to others. Such a conclusion can be drawn from global sensitivity analysis techniques that we discuss later in this paper. Figure 1(C) shows how the number of people who die immune to the disease changes as the ease at which the virus spread is increased. This robustness analysis reveals an exponential-like effect, where a further increase in the parameter value does not have much impact on the simulation response. Again this may be a key consideration when setting the value of this parameter.

Technique 2: Global Sensitivity Analysis using Latin-Hypercube Sampling

Using the latin-hypercube sampling technique, 500 sets of parameter values were generated over the specified parameter space. SPARTAN automatically creates Netlogo experiment XML files containing the chosen parameter values, and from these 500 sets of simulation responses were generated. Each parameter is then analyzed in turn, with output responses sorted by the value assigned to that parameter, and correlations identified as described previously.

Four of the automatically generated analysis plots can be seen in Figure 2. Each plot is for a particular parameter of interest, and a specified simulation response, both specified in the header. The objective is to identify any trends in simulation response between the value of this parameter and the response, although a subset of parameters are being perturbed simultaneously. Figure 2(A) reveals that there is no correlation between the duration of the infection and the number of people who die but are immune. In 2(B) on the other hand, there is a strong correlation between the number of people who die immune to the disease and the probability that they recover, as one would expect, although two other parameters are also being perturbed. This suggests that the researcher should give careful consideration to the value of

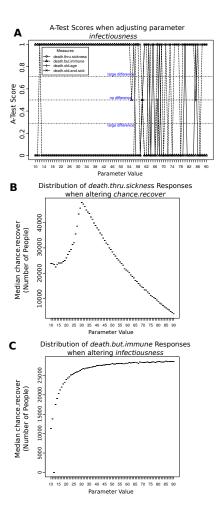


Figure 1: Exemplar robustness analysis output produced by SPARTAN. A: A-Test scores for Netlogo Virus simulation where the ease of catching the infection is perturbed (*infectiousness*). B: Distribution of the number of people who die through the virus when the chance of recovery is perturbed. C: Distribution of the number of people who die immune to the virus when *infectiousness* is perturbed

this parameter. Figures 2(C) and 2(D) are interesting as they reveal correlations between the parameter value and the simulation response for a subset of parameter values. In 2(C), the number of people who die through the infection is correlated with higher values for the chance of recovery, reinforcing the suggestion that this parameter is highly influential. In 2(D), the number of people who die through old age and are currently sick decreases rapidly under low infectious values, with no further trend from around 40% onwards. Such a result can be used to suggest that care is taken with the setting of this parameter, as it may not be influential on simulation response past 40%.

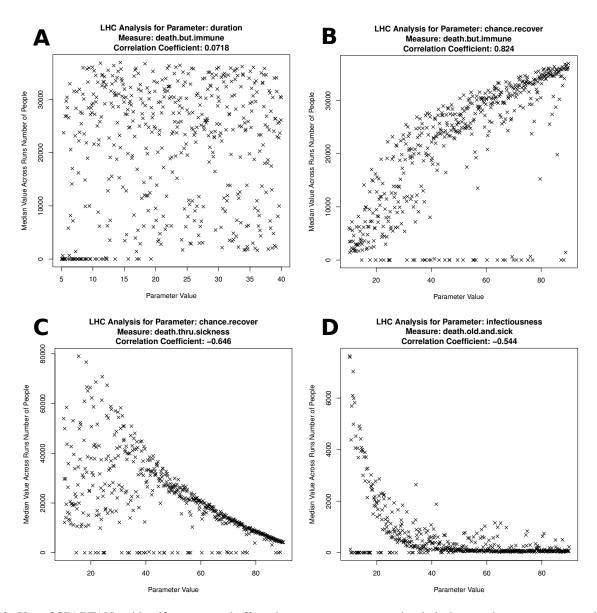


Figure 2: Use of SPARTAN to identify compound effects between parameters, using latin-hypercube parameter sampling. A: Number of people who die immune to the virus, sorted by the value assigned to duration of infection. B: Number of people who die immune to the virus, sorted by the value assigned to the chance they recover. C: Number of people who die through the virus, sorted by the value assigned to the chance they recover. D: Number of people who die through old age while they have the virus, sorted by the value assigned to *infectiousness*

Technique 3: Global Sensitivity Analysis using Extended Fourier Amplitude Sampling Test

Experiment parameters were generated using the sinusoidal curve sampling method detailed previously. With four parameters of interest (three in the simulation plus the 'dummy'), we have taken 65 parameter values from each frequency curve, and utilized three frequency phase shifts to ensure sufficient coverage of the parameter space, producing 780 parameter value sets. Similarly to Technique 2, SPARTAN automatically produces Netlogo experiment XML files for the chosen parameter values, and from these 780 simulation responses were generated.

The automatically generated graphs for two of the output measures can be seen in Figure 3. As the analysis focuses on each parameter individually, the graph shows the impact that each parameter has on a particular variance measure when that parameter is that of particular interest. The black bars are the Si value: the fraction of variance that can be explained by just that parameter when chosen as parameter of interest. The gray bars are the STi value: the fraction of output variance caused by higher-order non-linear affects between this parameter and the complementary parameters. The noticeable result from Figure 3 is that the STi value is high across all parameters for both the shown output measures. This suggests that there is a high degree of dependence between the parameters in this simulation. This is in contrast to our previously published example where this affect was relatively low (Alden et al., 2013). The interesting information in this graph comes from the black bars, the Si value. In Figure 3(A), showing the influence of each parameter on the number of people who die but are immune to the virus, the value of the infectiousness parameter sensitivity measure is noticeably higher than the other parameters, and the only parameter that is statistically significant in comparison to the 'dummy' measure, known to have no impact on the simulation. This supports earlier analyses that suggest that the *infectiousness* parameter value needs to be carefully considered. A similar conclusion is drawn from Figure 3(B), where again the *infectiousness* parameter is shown to have a statistically significant impact on the number of people who die through virus infection. The chance a person recovers from the infection is also determined to be statistically significant for this simulation response, as one would assume. Interestingly this analysis reveals that the duration of infection has little impact on variance in simulation response.

Discussion

Although a number of platforms have been developed that aid the development of simulations of complex systems, of which Netlogo is one, similar focus has not been given to the development of statistical tools that analyze the generated results. Platforms are beginning to offer enhanced parameter analysis through filtering and charting, MASS (MultiAgent Simulation Suite) being one example (Ivanyi et al.,

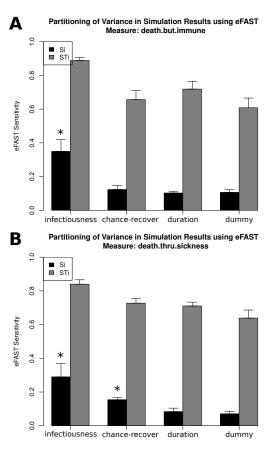


Figure 3: Use of eFAST method within SPARTAN to partition variance in simulation results between parameters.Black Bars: Si - the fraction of output variance explained by the value assigned to that parameter when parameter of interest; Gray Bars: STi - the variance caused by higher-order non-linear effects between that parameter and the others explored (includes value of Si). Error bars are standard error over three resample curves. A: Number of people who die immune to the virus. B: Number of people who die due to contracting the virus

2007), yet stress the importance of a link between the platform and a statistical environment (R,MATLAB, etc) for detailed analysis of exported data. Such a statistical analysis is vital for understanding the relationship between the simulation and the real world system it has been built to represent. We developed the SPARTAN package with the objective of providing researchers with the tools to examine this key relationship. Where researchers make the choice to utilize Netlogo as the software platform from which to design their simulation, the BehaviourSpace tool is very useful in providing a means of performing a sweep of potential parameter values. Yet this leaves the researcher to develop their own tools to analyze the simulation responses, to gain an understanding of the behavior of their simulation.

Here we have demonstrated our recent extension to SPARTAN that simplifies the process of performing both local and global sensitivity analyses of a Netlogo model. Through integrating Netlogo and SPARTAN, samples are generated that ensure full coverage of the parameter space, and Netlogo experiment XML files are created automatically. The researcher is now in the position to simply run the batch of experiments from the command line, and analyze the simulation responses using the relevant SPARTAN technique. Statistical information is produced in both graphical and data table forms, easing the identification of any effects generated through parameter perturbation. This provides the researcher with the key information required to gain a full understanding of how their simulation behaves, and where the key parameter pathways lie. To aid the adoption of this approach, we have ensured that SPARTAN is open source and freely available from both the CRAN package repository and our website (www.ycil.org.uk/software/spartan), and provided a detailed tutorial covering each technique. The sample model, data, and command line scripts are also available as exemplars. We see this as a useful addition to the simulation researcher's toolkit.

Acknowledgements

This work was part-funded by the Wellcome Trust [ref: 097829] through the Centre for Chronic Diseases and Disorders (C2D2) at the University of York. Jon Timmis is part funded by the Royal Society.

References

- Alden, K., Read, M., Timmis, J., Andrews, P. S., Veiga-Fernandes, H., and Coles, M. C. (2013). Spartan: A Comprehensive Tool for Understanding Uncertainty in Simulations of Biological Systems. *PLoS Computational Biology*, 9(2).
- Alden, K., Timmis, J., Andrews, P. S., Veiga-Fernandes, H., and Coles, M. C. (2012). Pairing experimentation and computational modelling to understand the role of tissue inducer cells in the development of lymphoid organs. *Frontiers in Immunology*, 3:1–20.
- Frank, B. and Baret, P. (2014). Simulating brown trout demogenetics in a river/nursery brook system: The individual-based model DemGenTrout. *Ecological Modelling*, 248:184–202.
- Helton, J. C. (2008). Uncertainty and sensitivity analysis for models of complex systems. In Barth, T. J., Griebel, M., Keyes, D. E., Nieminen, R. M., Roose, D., and Schlick, T., editors, Computational Methods in Transport: Verification and Validation, pages 207–228. Springer.
- Ivanyi, M., Bocsi, R., Gulyas, L., Kozma, V., and Legendi, R. (2007). The Multi-Agent Simulation Suite. In AAAI Fall Symposium Series, pages 56–63.
- Lee, T., Yao, R., and Coker, P. (2014). An analysis of UK policies for domestic energy reduction using an agent based tool. *Energy Policy*, 66:267–279.
- Luke, S. (2005). MASON: A Multiagent Simulation Environment. *Simulation*, 81(7):517–527.

- Marino, S., Hogue, I. B., Ray, C. J., and Kirschner, D. E. (2008). A methodology for performing global uncertainty and sensitivity analysis in systems biology. *Journal of theoretical biology*, 254(1):178–96.
- Patel, A., Harker, N., Moreira-Santos, L., Ferreira, M., Alden, K., Timmis, J., Foster, K. E., Garefalaki, A., Pachnis, P., Andrews, P. S., Enomoto, H., Milbrandt, J., Pachnis, V., Coles, M. C., Kioussis, D., and Veiga-Fernandes, H. (2012). Differential RET responses orchestrate lymphoid and nervous enteric system development. *Science Signalling*, 5(235).
- Read, M., Andrews, P. S., Timmis, J., and Kumar, V. (2012). Techniques for Grounding Agent-Based Simulations in the Real Domain: a case study in Experimental Autoimmune Encephalomyelitis. *Mathematical and Computer Modelling of Dynamical Systems*, 18(1):67–86.
- Saltelli, A. (2004). Sensitivity Analysis in practice: A guide to assessing scientific models. Wiley.
- Saltelli, A. and Bollardo, R. (1998). An alternative way to compute Fourier amplitude sensitivity test (FAST). *Comput. Stat. Data Anal.*, 26(4):445–460.
- Saltelli, A., Chan, K., and Scott, E. M. (2000). *Sensitivity Analysis*. Wiley series in probability and statistics Wiley.
- Thiele, J. C., Kurth, W., and Grimm, V. (2012). RNETLOGO: an R package for running and exploring individual-based models implemented in NETLOGO. *Methods in Ecology and Evolution*, 3(3):480–483.
- Vargha, A. and Delaney, H. D. (2000). A critique and improvement of the CL Common Language Effect Size Statistics of McGraw and Wong. *Journal of Educational and Behavioural Statistics*, 25:101–132.
- Wagner, N. and Agrawal, V. (2014). An agent-based simulation system for concert venue crowd evacuation modeling in the presence of a fire disaster. *Expert Systems with Applications*, 41(6):2817–2815.
- Wilensky, U. (1998). NetLogo Virus model. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.
- Wilensky, U. (1999). NetLogo itself. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.
- Yorke, J., Nathanson, N., Pianigiani, G., and Martin, J. (1979). Seasonality and the requirements for perpetuation and eradication of viruses in populations. *Journal of Epidemiology*, 109:103–123.

Hierarchical Heterogeneous Particle Swarm Optimization

Xinpei Ma^{1,2} and Hiroki Sayama^{1,2,3}

¹Collective Dynamics of Complex Systems Research Group

²Department of Bioengineering

³Department of Systems Science and Industrial Engineering

Binghamton University, State University of New York, Binghamton, NY, USA

xinpeima@gmail.com

Abstract

Particle swarm optimization (PSO) has recently been modified to several versions. Heterogeneous PSO is a recent extension which includes behavioral heterogeneity of particles. Here we propose a further developed version that has hierarchical interaction patterns among heterogeneous particles, which we call hierarchical heterogeneous PSO (HHPSO). Two algorithm designs that have been developed and tested are multi-layer HHPSO (ml-HHPSO) and multi-group HHPSO (mg-HHPSO). The performances of these algorithms were measured on a set of benchmark functions and compared with standard PSO and heterogeneous PSO. The results showed that the performances of both HHPSO algorithms were significantly improved from standard PSO and heterogeneous PSO, with higher quality of optimal solutions and faster convergence speed.

Particle swarm optimization (PSO), a population-based computational meta-heuristic, has been widely applied to many areas of science and engineering [6]. Many variants of PSO have demonstrated that the performance of PSO is largely influenced by particle behavior and interactions [1,5,7]. Among the recent extensions of PSO algorithms, introducing hierarchical social interaction [3] and particle heterogeneity [2] is our particular interest, because many biological collective systems seem to possess these properties to facilitate information exchange and balance different modes of collective behaviors (such as exploration and exploitation). To the extent of our knowledge, however, none of the earlier works combined hierarchical structures and heterogeneous behaviors in PSO.

Here we propose a *hierarchical heterogeneous PSO* (HHPSO) algorithm that integrates both heterogeneous particle behaviors and hierarchical interaction patterns into PSO. The basic idea of HHPSO is to introduce a two-phase procedure that first arranges particles into hierarchical structures based on their current fitness values and then assigns different behaviors to the particles based on their ranks in the hierarchy. To investigate the effects of varying hierarchical social structures and different levels of communication on PSO algorithms, we have designed two versions of HHPSO: *multi-layer* HHPSO (ml-HHPSO) and *multi-group* HHPSO (mg-HHPSO).

Multi-layer HHPSO is a basic version of HHPSO that focuses on communication between layers of particles (Fig. 1). In each iteration, particles are arranged into equally sized layers based on their current fitness values. Particles are allowed to interact with their immediate superior or inferior particles.

Compared to standard PSO where particles are attracted to global and personal best solutions, particles in ml-HHPSO are also attracted to immediately superior particles (except for particles in the top layer that are attracted to the ones with higher fitness within the same layer).

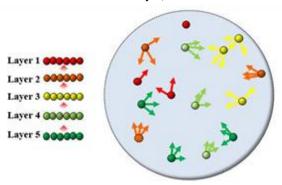


Figure 1: Schematic diagram of multi-layer HHPSO. Particles are structured into multiple layers based on their current fitness values; particles in each layer are attracted to those in the layer above (left). An illustrative example of particles in a search space is shown on the right, with attractive forces shown by arrows of different colors.

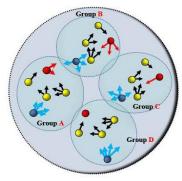


Figure 2: Schematic diagram of multi-group HHPSO. Particles are randomly split into multiple groups, inside each of which they are further structured into three layers based on their fitness values (red: head, blue: tail, yellow: body). Groups are spatially clustered in this figure just for visual clarity, but they can overlap in real cases. Body particles are attracted to others with higher fitness within its group (black arrows). Heads are attracted to other heads with higher fitness (red arrows). Tails are attracted to all heads (blue arrows).

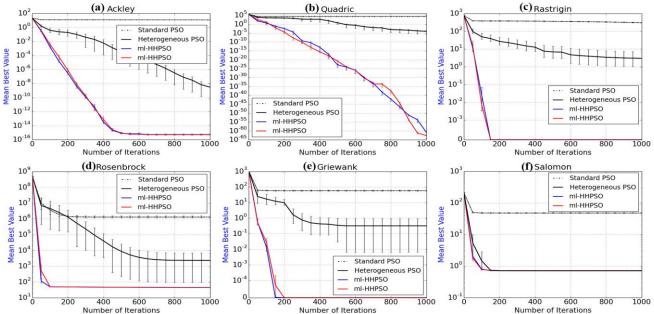


Figure 3: Average best fitness values of standard PSO, heterogeneous PSO, ml-HHPSO and mg-HHPSO for six benchmark functions [1]: (a) Ackley; (b) Quadric; (c) Rastrigin; (d) Rosenbrock; (e) Griewank; (f) Salomon. Error bars indicate the standard errors. Lower fitness means better results. Each swarm is made of 50 particles. Results are averaged over 50 independent simulation runs. (c) and (e) are plotted in a "symlog" scale of matplotlib [4] to show convergence to zero.

The second version of our algorithm, multi-group HHPSO, is a more elaborated one that aims to implement both vertical (i.e., between-layer) and horizontal (i.e., between-subpopulation) interaction of particles. In mg-HHPSO, a swarm is randomly separated into equally sized groups. Within each group, the particles are further separated into a three-layer hierarchical structure based on their current fitness values. Specifically, the particle with the highest fitness is designated as the *head*, the particle with the lowest as the tail, and the remaining particles as the body particles, each of which follows a different behavioral rule. Heads are responsible for leading other group members to where better solutions might exist. They are also attracted to other heads with higher fitness values. Body particles are attracted to their head and other body particles with higher fitness values, which contributes to an improvement in collective fitness. Tails are attracted to all heads regardless of which groups they belong to. This behavior of the tails helps increase the basic fitness level of the whole population from the bottom up. All particles are also attracted to global and personal bests, like in standard PSO.

Finally, in both ml- and mg-HHPSO, we implemented a mechanism for stagnancy detection, similar to the one used in [3]. Once stagnancy is detected, the stagnant particle randomly chooses a new behavioral rule rather than abides by the one determined by their ranks in the hierarchy.

The performance of these two HHPSO algorithms were evaluated through comparisons with standard PSO and heterogeneous PSO on six benchmark functions in a 50-dimensional search space [1] (Fig. 3). Five layers and ten groups were used in ml- and mg-HHPSO, respectively. Each algorithm was run 50 times for each benchmark function. Figure 3 shows that both of our HHPSO algorithms outperformed standard PSO and heterogeneous PSO. The quality of best solutions and convergence speed improved

significantly for most of the benchmark functions. In conclusion, we have proposed two versions of HHPSO algorithms which demonstrate significant improvement of search processes in both solution accuracy and convergence speed. We believe that this was because the combination of hierarchical structures and heterogeneous behaviors helped the swarm maintains diverse behaviors and effective communications among particles to balance its explorative and exploitative abilities simultaneously. Future research will concentrate on validating our algorithms using additional metrics and exploring other forms of implementing hierarchy and heterogeneity in more depth.

This material is based upon work supported by the US National Science Foundation under Grant No. 1319152.

References

- Bratton, D., & Kennedy, J. (2007, April). Defining a standard for particle swarm optimization. In Swarm Intelligence Symposium, 2007. SIS 2007. IEEE (pp. 120-127). IEEE
- [2] Engelbrecht, A. P. (2010). Heterogeneous particle swarm optimization. InSwarm Intelligence (pp. 191-202). Springer Berlin Heidelberg.
- [3] Janson, S., & Middendorf, M. (2005). A hierarchical particle swarm optimizer and its adaptive variant. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 35(6), 1272-1282.
- [4] Hunter, J., & Dale, D. (2007). The Matplotlib User's Guide. Matplotlib 0.90. 0 user's guide.
- [5] Kennedy, J., & Mendes, R. (2002). Population structure and particle swarm performance.
- [6] Parsopoulos, K. E., & Vrahatis, M. N. (2010). Particle swarm optimization and intelligence: advances and applications (pp. 1-4). Hershey: Information Science Reference.
- [7] Van den Bergh, F., & Engelbrecht, A. P. (2004). A cooperative approach to particle swarm optimization. Evolutionary Computation, IEEE Transactions on, 8(3), 225-239.

Constrained Group Counseling Optimization

Mohammad A. Eita¹, Amin B. Shoukry^{1,2} and Hitoshi C. Iba³

¹Comptuer Science and Engineering Department, Egypt-Japan University of Science and Technology (E-JUST), Alex, EGYPT

²Computer and Systems Engineering Department, Alexandria University, Alex, EGYPT

³Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan mohammad.eita@ejust.edu.eg

Abstract

Group Counseling Optimization (GCO) has recently been proposed in an attempt to emulate the human social behavior in solving life problems through counseling within a group. After its promising results in solving unconstrained singleobjective and multi-objective optimization problems, in this paper, GCO is extended to solve the constrained optimization problems for the first time. Also, a hybrid parameter-less constraint handling technique is proposed, which uses two wellknown constraint handling techniques: feasible rules and penalty function. The Constrained Group Counseling Optimization (CGCO) uses gradient-based mutation only in the case of all-equality-constraints COPs to reach the extremely small feasible region easily. Moreover, CGCO performance is tested by solving the constrained benchmarks function of the CEC 2010 competition. The results demonstrate that CGCO is competitive to other state-of-the-art algorithms and consistently reaches feasible solutions.

Introduction

Many real-world applications necessitate the solution of Constrained Optimization Problems (COPs). The solution of such problems means optimizing a given objective function while satisfying a set of imposed constraints. A COP can be defined as follows (Mezura-Montes and Coello C., 2011):

minimize
$$f(x)$$

subject to $g_j(x) \le 0$, $j = 1,...,q$ (1)
 $h_j(x) = 0$, $j = q + 1,...,m$
 $L_d \le x_d \le U_d$, $d = 1,...,D$

where $x = (x_1, x_2, ..., x_D) \in \mathbb{R}^D$ is a real-valued D-dimensional vector, f is the real valued objective function, g_j and h_j are q inequality constraints and (m-q) equality constraints, respectively, L_d and U_d are the lower and upper bounds of x_d , respectively.

To solve COPs, researchers used well-known natureinspired algorithms such as Particle Swarm optimization (PSO) (Kennedy et al., 1995), and Differential Evolution (DE) (Storn and Price, 1997), etc. Originally, all of these algorithms are mainly proposed to deal with unconstrained optimization problems so that it should add a Constraint Handling Technique (CHT) to enable them to deal with COPs.

Currently, seven categories of CHTs are known in the literature (Mezura-Montes and Coello C., 2011): Feasibility Rules (FR) (Deb, 2000), Stochastic Ranking (SR) (Runarsson and Yao, 2000), ε-constrained method (Takahama et al., 2005), Novel Penalty Functions, Novel special operators, Multi-objective concepts, Ensemble of constraint-handling techniques. In FR, a solution with less constraint violation is preferred. If two solutions have the same value of constraint violation, the fitter solution is preferred. In SR, a user-defined parameter called pf determines which criterion to use when comparing infeasible solutions: (1) based on their sum of constraints violation or (2) based only on their objective function values. In ε -constrained method, the value of $\varepsilon > 0$ relaxes the limit of considering a solution as feasible. In Novel Penalty Functions, researchers recently proposed two penalty-based approaches namely adaptive penalty function and dynamic penalty function. Adaptive penalty function (Tessema and Yen, 2009) calculates the penalty factor based on the status of the candidate solutions in the search space. Dynamic penalty functions (Tasgetiren and Suganthan, 2006) adopts the current generation number to decrease the penalty factor. In Novel special operators, operator such as boundary operator (Leguizamón and Coello C., 2009) is suggested. In Multi-objective concepts, a COP is turned into a bi-objective optimization problem (objective function and sum of constraints violation) (Wang et al., 2007). In ensemble of constraint-handling techniques (Mallipeddi and Suganthan, 2010a), more than one of the aforementioned categories are hybridized to get the advantages of each category.

For simplicity, the Constrained Group Counseling Optimization (CGCO) proposes the use of two parameter-less CHTs: FR and penalty function without any penalty factor. Gradient-based mutation operator (Takahama and Sakai, 2006) is also used to deal with COPs that have only equality constraints. CGCO is applied to a set of standard benchmark

COPs of the IEEE CEC 2010 competition (Mallipeddi and Suganthan, 2010b) and the results are compared with ε DEag (Takahama and Sakai, 2010), the winner of this competition, and another recently proposed algorithm Co-CLPSO (Liang et al., 2010).

This paper is organized as follows: section 2 outlines the related work. Section 3 provides an overview of GCO. Section 4 presents an overview of the CHTs and gradient-based mutation used by CGCO. Section 5 introduces the proposed CGCO. In section 6, CGCO is tested on COPs to evaluate its performance. Finally, the conclusions and future work are put forward in section 7.

Related Works

Here, some well-known nature-inspired algorithms, namely DE (Storn and Price, 1997), PSO (Kennedy et al., 1995) are briefly addressed.

In (Takahama and Sakai, 2006), an approach, called εDE, has been suggested to solve COPs using ε-constrained method as CHT and gradient-based mutation as a repair operator. Gradient-based mutation helps εDE handle COPs whose constraints are equality. This kind of COPs is difficult because the feasible region is very small. Afterwards, Takahama and Sakai added the concept of archive in their new approach εDEag (Takahama and Sakai, 2010). The archive increases the diversity of candidate solutions so that it gives better stability. In εDEag, a new controlling method of ε level is adopted. εDEag yielded promising results and was the winner of the CEC2010 competition (Mallipeddi and Suganthan, 2010b).

Liang et al. proposed an approach using PSO to solve COPs which is called cooperation comprehensive learning PSO (Co-CLPSO) (Liang et al., 2010). In Co-CLPSO, a novel CHT in which the population is divided into two subswarms is used. These two sub-swarms cooperate with each other in an attempt to solve the COPs. Particles of each swarm are responsible to deal with different constraints. Two swarms exchange their experiences to benefit each other. Sequential quadratic programming (SQP) is used as a local optimizer to improve the obtained solutions. Co-CLPSO ranked the fifth position in the CEC2010 competition.

Group Counseling Optimization

"Instead of mimicking the behavior of biological organisms such as birds, fish, ants, and bees, GCO is inspired by the human social behavior in solving life problems through *counseling within a group*" (Eita and Fahmy, 2010, 2014). Counseling (Burnard, 2002) is a well-established branch in sociology and psychology. This was the first time that a connection is found between population-based optimization and group counseling (Berg et al., 2006). Based on the group counseling concept, GCO is developed (Eita and Fahmy, 2010, 2014).

Four parameters affect the behavior of GCO:

- Number of group members acting as counselors, c, ($c \le m-1$).
 - Counseling probability, cp.
- Search range reduction coefficient, *red*, set into the range [0,1].
- \bullet Transition rate from the stage of exploration to that of exploitation, tr.

The GCO algorithm is illustrated in the following steps:

Step 1

At the very beginning, the algorithm initializes randomly a population with m D-dimensional candidate solutions X^i in the search space according to a beta distribution (Gentle, 2003); (Owen, 2008),

$$\beta(x) = \frac{x^{a-1}(1-x)^{b-1}}{B(a,b)} \quad 0 < x < 1$$

$$B(a,b) = \int_{0}^{1} t^{a-1}(1-t)^{b-1} dt \tag{2}$$

The case that both shaping parameters a and b are equal and less than unity, a=b<1, is adopted. According to the characteristics of beta distribution, the density function is symmetric and U-shaped. So, most of the candidate solutions lie near the search space boundaries. This implies that the global optimum is surrounded by these candidate solutions. Due to Other-members counseling strategy, as mentioned later in Step 3a, GCO moves from the search space boundaries to inside so that GCO most probably reaches global optimum. Shaping parameters a and b are often set to 0.1 as is the case in this paper.

Step 2

The objective function f(X) is evaluated for each solution X^i , producing m fitness values $f(X^i)$.

Step 3

For each solution X^i , an alternative solution X'^i is generated

$$X^{\prime i} = (x_1^{\prime i}, x_2^{\prime i}, ..., x_D^{\prime i}) \tag{3}$$

Each component $x_d^{\prime i}$ is obtained via one of two counseling strategies:

- (a) Other-members counseling
- (b) Self-counseling

A random number in the range [0,1] is generated, according to a uniform distribution, for each component x_d^i . It is named a *counseling decisive coefficient* (cdc). If $cdc \leq cp$, other-members counseling is used; otherwise, self-counseling is used. In the following, the production process of x_d^i , through these strategies, is demonstrated.

Step 3a: Other-members counseling $(cdc \le cp)$

Here, candidate solution X^i acts as a counselee. It asks for counseling of c other members, that act as counselors,

selected randomly out of the population. The value of x_d^{ij} is computed as *weighted* summation of the corresponding components (best experiences) of the c counselors. The weights are the contributions of the relevant counselors, in a brainstorming process. The weight, w_q , of component d of counselor q(q=1,2,...,c) is a random number in the range [0,1] according to a uniform distribution. Counselor q may be any member i. Eq. (4) constrains the summation of the c weights to unity,

$$\sum_{q=1}^{c} w_q = 1 \tag{4}$$

The component $x_d^{\prime i}$ is calculated as

$$x_d^{\prime i} = \sum_{q=1}^c w_q. x_d^{int_rand_q} \tag{5}$$

for all components of all candidate solutions. int_rand_q is an integer random number in the range [1,m], generated according to a uniform distribution. Eq. (5) indicates that c random numbers are generated for each component x_d^i . Also, $x_d^{int_rand_q}$ is the value of component d of counselor q being member int_rand_q . For each component, the set of c counselors differs. Furthermore, both w_q and int_rand_q varies for each value of i and d.

Step 3b: Self-counseling (cdc > cp)

An alternative component x_d^{i} is produced by searching around the current component x_d^{i} (best experience). The search range length $range\ length_d$ is computed as,

$$range_length_d = U_d - L_d \tag{6}$$

The searching process is achieved through modification of the component within a *reduced* range with length red. red is the search range reduction coefficient, set in the range [0,1]. We modify component d in a *maximum* range $[-mdf_max_d, mdf_max_d]$ about the current component, where

$$mdf_max_d = 0.5 \ red \ range_length_d$$
 (7)

The maximum modification value at the current iteration, $mdf_max_d^{itr}$, is calculated as

$$mdf_max_d^{itr} = mdf_max_d \left(1 - \frac{itr}{itr_max}\right)^{tr}$$
 (8)

where itr is the current iteration number, itr_max is the maximum number of iterations, tr is the transition rate at which GCO behavior transfers from exploration to exploitation. Eq. (8) indicates that $mdf_max_d^{itr}$ varies from mdf_max_d to zero. This means that when itr proceeds, GCO converts from exploration phase to exploitation phase.

For each component x_d^i , a random number is generated in the range $[-mdf_max_d^{itr}, mdf_max_d^{itr}]$ and then added to x_d^i to get the modified value $x_d^{\prime i}$ as follows

$$x_d^{\prime i} = x_d^i + rand_d^i (-mdf_max_d^{itr}, mdf_max_d^{itr})$$
 (9)

Step 4

The fitness value of $X^{\prime i}$, $f(X^{\prime i})$, is evaluated. If $f(X^{\prime i})$ is better than $f(X^i)$, then $X^{\prime i}$ replaces X^i ; otherwise, $X^{\prime i}$ is ignored and X^i is retained unchanged.

Repetition Steps (iterations)

Steps 3 and 4 are repeated until the stopping criterion is met.

Final Step

In the last iteration, GCO makes the decision by selecting the best solution out of m solutions.

Constrained Handling Techniques

In this paper, the sum of constraints violation of a solution x is defined as

$$CV(x) = \sum_{j=1}^{q} \max(0, g_j(x)) + \sum_{j=q+1}^{m} \max(0, |h_j(x)| - \delta)$$
(10)

where δ is a very small positive number. In the experimental section of this paper, δ is set to 0.0001.

Here, we address three CHTs used in our proposed algorithm: namely FR, parameter-less penalty function, and gradient-based mutation.

Feasibility Rules (FR)

In (Deb, 2000), the author suggested FR as a CHT. FR is popular in this field because it can be added to any nature-inspired algorithm without adding any new parameter (parameter-less). To compare two solutions, FR employs three feasibility criteria as follows: (1) for two feasible solutions, the solution with the better objective function is preferred, (2) for a feasible and an infeasible solution, the feasible solution is preferred, (3) for two infeasible solutions, the solution with the lower sum of constraint violation is preferred.

Parameter-less Penalty Function

Penalty function transforms COPs to unconstrained optimization problems which are easy to deal with. Due to its simple implementation, penalty function technique is popular and it is used by a lot of researchers.

In (Debchoudhury et al., 2013), the authors used the simple penalty function without any penalty factor that needs to be adjusted, as follows

$$\phi(x) = f(x) + CV(x) \tag{11}$$

where f(x) is the objective function, CV(x) is the sum of constraints violation, $\phi(x)$ is the modified objective function.

Gradient-based Mutation

Gradient-based mutation (Takahama and Sakai, 2006, 2010) is a repair operator, by which an infeasible solution is converted to a feasible solution. The constraint functions vector C(x) and the constraint violation vector $\Delta C(x)$ are as follows.

$$C(x) = [g_1(x), ..., g_q(x), h_{q+1}(x), ..., h_m(x)]^T$$
 (12)

$$\Delta C(x) = [\Delta g_1(x), ..., \Delta g_q(x), h_{q+1}(x), ..., h_m(x)]^T$$
 (13)

where $\Delta g_i(x) = max(0, g_i(x))$.

The linear approximation of the constraint violation vector $\Delta C(x)$ through Taylor expansion is defined as follows,

$$\nabla C(x) \, \Delta x = -\Delta C(x) \tag{14}$$

$$\Delta x = -\nabla C(x)^{-1} \, \Delta C(x) \tag{15}$$

where Δx is the change in x required to make solution x feasible, $\nabla C(x)$ is the gradient of the constraint functions which is numerically approximated as follows:

$$\nabla C(x) = \frac{1}{\eta} [C(x + \eta e_1) - C(x), ..., C(x + \eta e_n) - C(x)]$$
 (16)

where η is a very small number, e_i is the unit vector in the direction of dimension i. Generally, $\nabla C(x)$ matrix is not square, so the pseudo inverse (Campbell and Meyer, 2009) is the best numerical approximation of $\nabla C(x)^{-1}$. The new value of x is calculated by the following formula,

$$x^{new} = x^{old} + \Delta x \tag{17}$$

Proposed CGCO

After GCO gave promising results in solving unconstrained single-objective (Eita and Fahmy, 2010, 2014) and multi-objective function (Ali and Khan, 2013), GCO is extended, here, to solve COPs. In the CGCO algorithm, FR and parameter-less penalty function are hybridized as a proposed novel CHT. The hybridization is adopted to gain the advantages of FR and parameter-less penalty function and to discard their disadvantages. **Table 1** summarizes both the advantages and disadvantages of various CHTs (Mani and Patvardhan, 2009). It should be obvious that in addition to the merits of the penalty function technique, the parameter-less penalty function does not need to adjust any penalty factor.

Table 1: Comparison Among Various CHTs (Mani and Patvardhan, 2009)

Item	Gradient	Penalty	Parameter-less	FR	Proposed
	based	Function	Penalty		CHT
	mutation		Function		
Computationally expensive	Yes	No	No	No	No
More feasible solution wins	Yes	No	No	Yes	Yes
Exploration	Good	Good	Good	Poor	Good
Parameter-less	No	No	Yes	Yes	Yes
Simple implementation	No	Yes	Yes	Yes	Yes

CGCO possesses a good exploration capability, avoids premature convergence of FR (Mezura-Montes and Coello C., 2011), using the parameter-less penalty function and can pick the more feasible solutions using FR and stores it into *X* .*FR* . *X* .*FR* replaces a solution from the population other than the best solution according to modified objective function so that evolution process benefits from the more feasible solution obtained by FR.

Gradient-based mutation is a computationally expensive repair operator. When the feasibility region is extremely small as is the case in COPs which have only equality constraints, gradient-based mutation helps iteratively the CGCO to reach feasibility. Due to the nonlinearity of the constraints and the approximation given in Eq.(14), gradient-based mutation is applied to a solution iteratively to increase the feasibility of the solution. *N* is the maximum number of iterations of gradient-based mutation.

Algorithms 1 and 2 describe the pseudo code of the CGCO algorithm and gradient-based mutation operator respectively.

Experiments and Results

In this section, the CGCO performance is tested through solving 18 10-D and 18 30-D constrained benchmark functions of IEEE CEC 2010 competition (Mallipeddi and Suganthan, 2010b). Then the comparison is made with two competitive algorithms: ε DEag (Takahama and Sakai, 2010) and Co-CLPSO (Liang et al., 2010). ε DEag and Co-CLPSO ranked the first and the fifth position in this competition, respectively. For each function, CGCO is run for 25 independent runs. The algorithm is coded in C++, and run on a PC with Windows 7 OS, a 4 GB Ram, Intel 2.67 GHz core i7 processor. The stopping criterion is the maximum number of function evaluations FE_{max} . For 10-D functions, FE_{max} is equal to 200,000 FE_s . For 30-D functions, FE_{max} is equal to 600,000 FE_s .

CGCO uses the following parameters: Population size m = 40, c = 2, cp = 0.01 (this value has been experimen-

Algorithm 1 The CGCO algorithm, where FE is the number of current function evaluation, u(a,b) is a uniform random number in [a,b]

- 1. Initialization of the population of solutions (X^1,\ldots,X^m) according to beta distribution
- 2. For each solution, evaluation of objective function $f(X^i)$, sum of constraint violation $CV(X^i)$, modified objective function $\phi(X^i)$
- 3. According to FR, determining the index of the best solution r, and assigned X^r to $X FR (X FR = X^r)$
- 4. repeat{

5. **for**(
$$i = 1; i < m; i + +)$$
{

6. **for**(
$$j = 1; j \le D; j + +)$$
{

- 7. cdc = u(0,1)
- 8. **if**(cdc < cp){
- 9. Excluding solution i, select randomly different c solutions

10.
$$x_d^{\prime i} = \sum_{q=1}^c w_q. x_d^{int_rand_q}$$

- 11. }else-
- 12. $mdf_max_d^{FE} = mdf_max_d \left(1 \frac{FE}{FE\ max}\right)^{tr}$
- 13. **repeat**{
- 14. $x_d^{ii} = x_d^i + rand_d^i(-mdf_max_d^{FE}, mdf_max_d^{FE})$
- 15. $\{ \mathbf{until}(x_d^{\prime i} \text{ inside the allowable range }) \}$
- 16.
- 17. Evaluation of $f(X^{\prime i}), CV(X^{\prime i}), \phi(X^{\prime i})$; FE = FE + 1
- 18. **if** $(\phi(X'^i) < \phi(X^i))X^i = X'^i$
- 19. According to FR, **if** $(X^{\prime i})$ better than $X FR = X^{\prime i}$
- 20. **if**(all constraints are equality && $\phi(X^i)$ is the best value){
- 21. Doing gradient-based mutation to $X^{\prime i}$ as in **Algorithm 2**
- 22. }
- 23.
- 24. Determining the index of the best solution of the population according to modified objective function, *best*
- 25. **if**(r == best){ // Injection of $X \perp FR$ into the population
- 26. Generating randomly a new value of $r \in [1, m]$ such that $r \neq best$
- 27. $X^r = X_F R$
- 28. }else{
- 29. $X^r = X \mathcal{F} R$
- 30. }
- 31. $\}$ **until**(FE == FEmax)
- 32. According to FR, comparing($X ext{-}FR, X^{best}$) and output the best

Algorithm 2 Gradient-based mutation to $X^{\prime i}$

- 1. **for**(n = 1; X'^i is infeasible && $n \le N$; n + +)
- 2. $\Delta X^{\prime i} = -\nabla C(X^{\prime i})^{-1} \Delta C(X^{\prime i})$
- 3. **if**($X'^i + \Delta X'^i$ is inside the allowable range){
- 4. $X^{\prime i} = X^{\prime i} + \Delta X^{\prime i}$
- 5. }else{
- 6. Exit from the loop
- 7.
- 8. Evaluation of $f(X'^i)$, $CV(X'^i)$, $\phi(X'^i)$
- 9. **if**($\phi(X'^i) < \phi(X^i)$){ $X^i = X'^i$ }
- 10. According to FR, if (X'^i) better than X_FR $\{X_FR = X'^i\}$
- 11. FE = FE + D + 1
- 12. }

tally found suitable for all functions except 30-D C01&C02 for which the value 0.1 has given better performance), red = 0.5, tr = 7, N = 100. The results and feasibility rate for 10-D and 30-D are given in **Appendix A**.

From **Appendix A**, it is significant to mention that the feasibility rate of CGCO is 100% for all 18 10-D and 18 30-D test functions. On the other hand, the feasibility rate of εDEag is 100% for only 35 test functions and 12% for 30-D C12. Also, the feasibility rate of Co-CLPSO is 100% for only 31 test functions and 0% for 10-D C11, 0% for 30-D C03, 80% for 30-D C04, 0% for 30-D C11, and 92% for 30-D C12. Also, **Appendix A** shows that εDEag results are slightly better than CGCO results for only 10-D test functions.

Table 2 outlines the comparison of CGCO vs. εDEag and Co-CLPSO. Wilcoxon Signed Rank Test for 5% significance level is performed. Three symbols (+, -, and =) are used in this respect. The symbol "+" denotes that the first algorithm is significantly better than the second one, the symbol "-" indicates that the first algorithm is significantly worse than the second, and the symbol "=" means that there is no significant difference between the two algorithms. The table shows that, for 10-D functions, CGCO is competitive to Co-CLPSO. But for 30-D functions, CGCO is competitive to εDEag and outperforms Co-CLPSO based on the average value. This case is expected according to no free lunch theorem. Here, it is important to report the following points: 1) CGCO does not use any local optimizer and depends only on its behavior to reach the optimum, while Co-CLPSO uses a local optimizer called Sequential Quadratic Programming (SQP) to improve the quality of the solutions (Liang et al., 2010). 2) CGCO does not use any archive, while Table 2: COMPARISON OF CGCO vs. εDEag and Co-CLPSO

Dim	Algorithms	Objective	Better	Equal	Worse	Test
		Function				
10 D	CGCO vs. εDEag	Best	3	3	12	-
		Average	2	2	14	-
	CGCO vs. Co-CLPSO	Best	5	0	13	=
		Average	10	0	8	=
30 D	CGCO vs. EDEag	Best	11	0	7	=
	CGCO vs. EDEag	Average	9	0	9	=
	CGCO vs. Co-CLPSO	Best	9	0	9	=
		Average	14	0	4	+

εDEag uses an archive to preserve the diversity of the solutions (Takahama and Sakai, 2010). 3) CGCO employed the gradient-based mutation only for 14 all-equality-constraints COPs (C03, C04, C05, C06, C09, C10, and C11) with 10-D and 30-D, while εDEag employed the gradient-based mutation for all 36 test functions (Takahama and Sakai, 2010). 4) CGCO adopts a parameter-less CHT for 22 test functions and needs to set only one parameter 'N' of gradient-based mutation for 14 all-equality-constraints COPs, while εDEag needs to set 4 parameters for its CHT (2 parameters for ε-constrained method and 2 parameters for gradient-based mutation) (Takahama and Sakai, 2010).

Conclusions

Group Counseling optimization (GCO) has basically been introduced to solve unconstrained optimization problems. In this paper, the Constrained Group Counseling Optimization (CGCO) is presented to solve COPs. CGCO adopts a new parameter-less CHT in which hybridization between FR and penalty function techniques is implemented. In addition to this, gradient-based mutation is used in the case of COPs that have only equality constraints so that the feasible solution is obtained quickly. The proposed algorithm is applied to the constrained benchmark functions of the CEC 2010 competition. A comparison is made between CGCO and two well-known algorithms. The results show that the performance of CGCO is competitive and the feasibility rate is always 100% for all functions.

For future work, the behavior of CGCO needs to be studied with other CHTs such as SR and ϵ -constrained method. The applicability of the CGCO to real-world applications should be, also, investigated.

Acknowledgment

This research is supported by the Ministry of Higher Education (MoHE) of Egypt through PhD fellowships. Our sincere thanks to E-JUST University for guidance and support.

References

- Ali, H. and Khan, F. A. (2013). Group counseling optimization for multi-objective functions. In *Evolutionary Computation* (CEC), 2013 IEEE Congress on, pages 705–712. IEEE.
- Berg, R. C., Landreth, G. L., and Fall, K. A. (2006). *Group Counseling: Concepts and Procedures Fourth Edition*. Routledge.
- Burnard, P. (2002). Practical counselling and helping. Routledge.
- Campbell, S. L. and Meyer, C. D. (2009). *Generalized inverses of linear transformations*, volume 56. SIAM.
- Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering*, 186(2):311–338.
- Debchoudhury, S., Biswas, S., Kundu, S., Das, S., Vasilakos, A. V., and Mondal, A. (2013). Modified estimation of distribution algorithm with differential mutation for constrained optimization. In *Evolutionary Computation (CEC)*, 2013 IEEE Congress on, pages 1724–1731. IEEE.
- Eita, M. A. and Fahmy, M. M. (2010). Group counseling optimization: A novel approach. In *Research and Development in Intelligent Systems XXVI*, pages 195–208. Springer.
- Eita, M. A. and Fahmy, M. M. (2014). Group counseling optimization. *Applied Soft Computing*.
- Gentle, J. E. (2003). Random number generation and Monte Carlo methods. Springer.
- Kennedy, J., Eberhart, R., et al. (1995). Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks*, volume 4, pages 1942–1948. Perth, Australia.
- Leguizamón, G. and Coello C., C. A. (2009). Boundary search for constrained numerical optimization problems with an algorithm inspired by the ant colony metaphor. *Evolutionary Computation, IEEE Transactions on*, 13(2):350–368.
- Liang, J. J., Zhigang, S., and Zhihui, L. (2010). Coevolutionary comprehensive learning particle swarm optimizer. In *Evolutionary Computation (CEC)*, 2010 IEEE Congress on, pages 1–8. IEEE.
- Mallipeddi, R. and Suganthan, P. N. (2010a). Ensemble of constraint handling techniques. *Evolutionary Computation, IEEE Transactions on*, 14(4):561–579.
- Mallipeddi, R. and Suganthan, P. N. (2010b). Problem definitions and evaluation criteria for the cec 2010 competition and special session on single objective constrained real-parameter optimization. Nangyang Technological University, Singapore.
- Mani, A. and Patvardhan, C. (2009). A novel hybrid constraint handling technique for evolutionary optimization. In *Evolutionary Computation*, 2009. CEC'09. IEEE Congress on, pages 2577–2583. IEEE.
- Mezura-Montes, E. and Coello C., C. A. (2011). Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194.

- Owen, C. (2008). Parameter estimation for the beta distribution. Master's thesis, Brigham Young University, Utah, USA.
- Runarsson, T. P. and Yao, X. (2000). Stochastic ranking for constrained evolutionary optimization. *Evolutionary Computation, IEEE Transactions on*, 4(3):284–294.
- Storn, R. and Price, K. (1997). Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359.
- Takahama, T. and Sakai, S. (2006). Constrained optimization by the &# 949; constrained differential evolution with gradientbased mutation and feasible elites. In *Evolutionary Computation*, 2006. CEC 2006. IEEE Congress on, pages 1–8. IEEE.
- Takahama, T. and Sakai, S. (2010). Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. In *Evolutionary Computation* (*CEC*), 2010 IEEE Congress on, pages 1–9. IEEE.
- Takahama, T., Sakai, S., and Iwane, N. (2005). Constrained optimization by the ε constrained hybrid algorithm of particle swarm optimization and genetic algorithm. In AI 2005: Advances in Artificial Intelligence, pages 389–400. Springer.
- Tasgetiren, M. F. and Suganthan, P. N. (2006). A multi-populated differential evolution algorithm for solving constrained optimization problem. In *Evolutionary Computation*, 2006. CEC 2006. IEEE Congress on, pages 33–40. IEEE.
- Tessema, B. and Yen, G. G. (2009). An adaptive penalty formulation for constrained evolutionary optimization. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 39(3):565–578.
- Wang, Y., Cai, Z., Guo, G., and Zhou, Y. (2007). Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 37(3):560–575.

 ${\bf Appendix~A}$ FUNCTION VALUES ACHIEVED BY CGCO, $\epsilon DEag,$ Co-CLPSO FOR THE CEC2010 TEST PROBLEMS

D 11		10D					30D			
Problem	Algorithm	Feasibility Rate	Best	Mean	SD	Feasibility Rate	Best	Mean	SD	
	CGCO	100	-7.473104E-01	-7.452169E-01	4.991886E-03	100	-8.218384E-01	-8.189027E-01	4.635651E-03	
C01	εDEag	100	-7.473104E-01	-7.470402E-01	1.323339E-03	100	-8.218255E-01	-8.208687E-01	7.103893E-04	
	Co-CLPSO	100	-7.4731E-01	-7.3358E-01	1.7848E-02	100	-8.0688E-01	-7.1598E-01	5.0252E-02	
	CGCO	100	-2.277704E+00	-2.246991E+00	1.264116E-02	100	-2.214588E+00	-2.158273E+00	5.783516E-02	
C02	εDEag	100	-2.277702E+00	-2.269502E+00	2.3897790E-02	100	-2.169248E+00	-2.151424E+00	1.197582E-02	
	Co-CLPSO	100	-2.2777	-2.2666	1.4616E-02	100	-2.2809	-2.2029	1.9267E-01	
C03	CGCO	100	3.032171E-06	2.737753E-05	1.542211E-05	100	2.540517E-03	4.620062E+00	1.072924E+01	
	εDEag	100	0.00E+00	0.00E+00	0.00E+00	100	2.867347E+01	2.883785E+01	8.047159E-01	
	Co-CLPSO	100	2.4748E-13	3.5502E-01	1.77510	0	N/A	N/A	N/A	
	CGCO	100	-9.989973E-06	-8.374013E-06	1.031334E-06	100	-1.777536E-07	5.145977E-06	3.897395E-06	
C04	εDEag	100	-9.992345E-06	-9.918452E-06	1.54673E-07	100	4.698111E-03	8.162973E-03	3.067785E-03	
	Co-CLPSO	100	-1.00E-05	-9.3385E-06	1.0748E-06	80	-2.93E-06	1.1269E-01	5.6335E-01	
	CGCO	100	-4.836106E+02	-4.836106E+02	0.00E+00	100	-4.764488E+02	-4.508943E+02	1.445262E+01	
C05	εDEag	100	-4.836106E+02	-4.836106E+02	3.89035E-13	100	-4.531307E+02	-4.495460E+02	2.899105E+00	
	Co-CLPSO	100	-4.8361E+02	-4.8360E+02	1.9577E-02	100	-4.8360E+02	-3.1249E+02	8.8332E+01	
	CGCO	100	-5.786585E+02	-5.786703E+02	7.433578E-02	100	-5.208747E+02	-4.462454E+02	6.149371E+01	
C06	εDEag	100	-5.786581E+02	-5.786528E+02	3.6271690E-03	100	-5.285750E+02	-5.279068E+02	4.748378E-01	
	Co-CLPSO	100	-5.7866E+02	-5.7866E+02	5.7289E-04	100	-2.8601E+02	-2.4470E+02	3.9481E+01	
	CGCO	100	1.344985E-07	2.703616E+00	1.454563E+00	100	9.906752E+00	2.302078E+01	1.368257E+01	
C07	εDEag	100	0.00E+00	0.00E+00	0.00E+00	100	1.147112E-15	2.603632E-15	1.233430E-15	
207	Co-CLPSO	100	1.0711E-09	7.9732E-01	1.6275	100	3.7861E-11	1.1163	1.8269	
	CGCO	100	1.245218E-02	7.161994E+00	3.581421E+00	100	1.799808E+01	2.286629E+01	2.142798E+00	
C08		100	0.00E+00	6.727528E+00	5.560648E+00	100	2.518693E-14	7.831464E-14	4.855177E-14	
CUS	εDEag Co-CLPSO	100				100				
			9.6442E-10	6.0876E-01	1.4255		4.3114E-14	4.7517E+01	1.1259E+02	
Goo	CGCO	100	7.646936E-16	3.614285-08	6.843696E-08	100	3.888829E-09	1.25734E+01	3.2612E+01	
C09	εDEag	100	0.00E+00	0.00E+00	0.00E+00	100	2.770665E-16	1.072140E+01 1.4822E+08	2.821923E+01	
	Co-CLPSO	100	3.7551E-16	1.9938E+10	9.9688E+10	100	1.9695E+02		2.4509E+08	
a	CGCO	100	1.466726E-07	3.403369E+01	1.580944E+01	100	3.146641E+01	3.414345E+01	1.895779E+00	
C10	εDEag	100	0.00E+00	0.00E+00	0.00E+00	100	3.252002E+01	3.326175E+01	4.545577E-01	
	Co-CLPSO	100	2.3967E-15	4.9743E+10	2.4871E+11	100	3.1967E+01	1.3951E+09	5.8438E+09	
	CGCO	100	-1.516158E-03	-1.047986E-03	6.982165E-04	100	-3.800417E-04	-3.754486E-04	3.625592E-06	
C11	εDEag	100	-1.52271E-03	-1.52271E-03	6.3410350E-11	100	-3.268462E-04	-2.863882E-04	2.707605E-05	
	Co-CLPSO	0	N/A	N/A	N/A	0	N/A	N/A	N/A	
C12	CGCO	100	-1.973112E-01	-1.221012E-01	9.193794E-02	100	-1.983111E-01	8.844418E-02	5.089598E-01	
	εDEag	100	-5.700899E+02	-3.367349E+02	1.7821660E+02	12	-1.991453E-01	3.562330E+02	2.889253E+02	
	Co-CLPSO	100	-1.2639E+01	-2.3369	2.4329E+01	92	-1.9926E-01	-1.9911E-01	1.1840E-04	
C13	CGCO	100	-6.842937E+01	-6.842936E+01	4.627696E-06	100	-6.842914E+01	-6.83411E+01	2.380819E-01	
	εDEag	100	-6.842937E+01	-6.842936E+01	1.02596E-06	100	-6.642473E+01	-6.535310E+01	5.733005E-01	
	Co-CLPSO	100	-6.842936E+01	-6.397445E+01	2.13408E+00	100	-6.2752E+01	-6.0774E+01	1.1176	
C14	CGCO	100	1.790238E-01	3.734062E+00	1.164704E+00	100	1.800895E+01	2.566797E+01	1.331492E+01	
	εDEag	100	0.00E+00	0.00E+00	0.00E+00	100	5.015863E-14	3.089407E-13	5.608409E-13	
	Co-CLPSO	100	5.78E-12	3.1893E-01	1.1038	100	3.28834e-09	0.0615242	0.307356	
	CGCO	100	1.800656E-02	3.155997E+00	1.253508E+00	100	2.08679E+01	2.365358E+01	1.199714E+00	
C15	εDEag	100	0.00E+00	1.798980E-01	8.813156E-01	100	2.160345E+01	2.160376E+01	1.104834E-04	
	Co-CLPSO	100	3.0469E-12	2.9885	3.3147	100	5.7499E-12	5.1059E+01	9.1759E+01	
	CGCO	100	5.281739E-02	5.37987E-01	2.417903E-01	100	1.063727E+00	1.077787E+00	1.077612E-02	
C16	εDEag	100	0.00E+00	3.702054E-01	3.7104790E-01	100	0.00E+00	2.168404E-21	1.062297E-20	
	Co-CLPSO	100	0.00E+00	5.9861E-03	1.3315E-02	100	0.00E+00	5.2403E-16	4.6722E-16	
C17	CGCO	100	1.395648E-11	2.498691E-03	3.819487E-03	100	5.2054121E-02	3.693843E-01	1.766043E-01	
	εDEag	100	1.46318E-17	1.249561E-01	1.937197E-01	100	2.165719E-01	6.326487E+00	4.986691E+00	
	Co-CLPSO	100	7.6677E-17	3.7986E-01	4.5284E-01	100	1.5787E-01	1.3919	4.2621	
C18	CGCO	100	1.984565E-20	3.57925E-14	8.240038E-14	100	3.22061E-02	2.726909E+00	3.697505E+00	
	εDEag	100	3.73144E-20	9.678765E-19	1.811234E-18	100	1.226054E+00	8.754569E+01	1.664753E+02	
C10 ,										

Part III Poster Presentations

Causes vs Benefits in the Evolution of Prey Grouping

Ritwik Biswas^{1,4}, Charles Ofria^{1,2,3}, David M. Bryson¹ and Aaron P. Wagner¹

BEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, MI 48824, USA
 Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA
 Program in Ecology, Evolutionary Biology, and Behavior, Michigan State University, East Lansing, MI 48824, USA
 College of Engineering, The University of Michigan, Ann Arbor, MI 48109, USA

Abstract

The presence of predators alters the evolutionary pressures acting on prey populations, often driving them to engage in new behaviors in order to avoid being the target of an attack. One common antipredator behavior is group formation, which can reduce the odds of any individual prey being the target of a given attack, typically scaling inversely with the number of prey in the group. This "dilution effect" is often hypothesized to be the primary driver of prey group formation as an anti-predator strategy. However, groups may help with predator avoidance in other ways as well. For example, prey behaviors or physical characteristics that visually confuse predators may reduce their ability to target an individual and make a kill. Indeed, some have suggested that the "predator confusion" effect alone is sufficient to drive the evolution of grouping in prey. Here we examine coevolving populations of predators and prey using the Avida digital evolution platform. We evaluate the relative importance of these two potential drivers of the evolution of prey grouping and show that the dilution effect, an inherent property of most prey groups, readily creates the pressures necessary for the evolution of prey grouping. In contrast, we found no evidence that predator confusion plays a significant role in prey group formation. Instead, the dilution effect alone is indicated as the primary driver of antipredator prey grouping strategies.

Introduction

Organism clustering is recurrent in nature, as is demonstrated by diverse species, such as fish, cattle, and bees (Patridge 1982. Omholt 1987). Much literature has dealt with the environmental factors and selective pressures favoring such strategies (Jakobsen et al 1988, Mooring et al 1992). For example, bees huddle in high-density groups in order to conserve heat energy and ultimately increase their lifespan (Nagy et al 1976). However, clustering is also hypothesized to be an evolutionary favored strategy in the context of predation avoidance (Kunz et al 2006), and is sometimes considered the most influential facilitator of the evolution and expression of cooperative and grouping behaviors in prey populations (Krams 2010). From the perspective of an individual prey, spatial grouping reduces the chance of being the target of a predator attack (Turner et al 1986). This simple dilution effect is thought to be one of the primary selective pressures favoring the evolution grouping in prey, a hypothesis supported by empirical models (Foster 1981, Turner et al 1986).

Although dilution is believed to be a dominant force favoring prey grouping, additional factors such as probability of detection, coordinated evasion, or predator confusion could play additional, potentially critical, roles (Reynolds 1987). With respect to the last, clustering by prey is often argued to confuse predators as to the nature of the group and the individuals in it (Kunz et al. 2006, Cosner et al. 1999, Jeschke and Tollrian 2007). Additionally, prey may exhibit confusing behavioral or morphological traits (Relyea 2001), e.g. rapid and erratic changes in directions or confusing flashes of color. The evolution of these confusion traits may reduce the overall odds of a predator being able to successfully target individuals in the group for an attack. As such, others have suggested that predator confusion creates selective pressure for prey grouping (Millinksi 1979).

While there is ample evidence indicating that confusion traits can benefit individuals already living in groups, it is not clear whether those benefits were necessary for the evolution of grouping strategies. This uncertainty arises for several reasons: (1) The dilution effect is an inherent property of most prey groups (Foster 1981, Delm 1990). It does not require the evolution of additional behavioral or morphological traits in order to come into play. By contrast, the confusion effect relies on the existence of additional prey traits that confuse predators once prey are in groups (Krakauer 1995, Millinksi 1984). Thus it would be a large evolutionary leap for confusion to be the factor initially favoring grouping. (2) Predators evolve sophisticated sensory and behavioral mechanisms for detecting and pursuing prey (Bengtson 2002, Abrams 2000). If predators and prev are locked in perceptionconfusion arms-races (Dawkins et al 1979), we would expect predators to evolve traits to counter confusion effects, thus weakening their impact over time. The strength of the dilution effect, however, is dependent only on the number of prey in a group and the number of predators attacking. Thus, the strength of its effect should be expected to remain stable over evolutionary time. (3) The confusion effect presupposes that predators can be confused by prey behaviors (Landeau et al 1986, Millinksi 1979, Krakuer 1995). E.g., whales are unlikely to be dependent on the sorts of traits that fish might exhibit to confuse predators closer to their own body size. Equally, blind predators will not be confused by visual deceptions. Dilution, however, is not dependent on any such

close mapping between predator and prey traits, and thus should be a factor more commonly at play. (4) To date, no studies have evaluated the effects of confusion on the evolution of grouping strategies without removing the effects of dilution. E.g., while Kunz et al (2006; similarly also Olson et al. 2013) showed that confusion is sufficient to evolve grouping in prey, they show this only for clonal groups – a rare and specialized circumstance which has the side effect of removing the dilution effect.

In order to understand what conditions push prey to evolve and exhibit grouping behaviors, we tested for the effects of both dilution and predator confusion in a coevolving predatorprey system. We show that while the dilution effect readily promotes the evolution of prey grouping, no mechanism of confusion had any impact on clustering behavior. Note that we discuss these traits in the context of prey 'grouping', 'clustering', or 'herding'. We consider such life strategies to be distinct from 'swarming', which is a more specialized behavior that is explicitly responsive to behavioral stimuli such as the presence of predators (Okubo 1986) or expectations of finding food (Salge & Polani 2011). That is, hyenas live in groups (Karanth et al 2000), but clearly don't swarm. Wildebeest live in groups, not swarms, but may swarm when attacked (Silk 2007, Gueron et al 1993). Bees live in groups and swarm to change nests (Beekman & Ratnieks 2000. Boreham & David 1987). Some fruit flies swarm to find mates, but live life alone (Sivinski et al. 1997). Because other studies evaluating the confusion effect have demonstrated grouping as a life-history strategy (Kunz 2006, Olson et al. 2013), not as a behavioral response to stimuli, we focus on factors favoring the evolution of grouping.

Methods

We used the digital evolution software platform Avida (Ofria et al. 2009; Bryson & Ofria 2013) to evaluate the effects of dilution and confusion on the evolution of prey grouping as an anti-predator behavior. Avida is valuable for this kind of work in carrying the main benefits of simulations, for example, rapid generation times and full control over configuration options. However, unlike in population genetic simulators and genetic algorithms, evolution in Avida is unrestricted and unguided, not requiring the use of explicit selection functions to impose fitness values on individuals. Instead, Avida is utilizes natural selection, with an organism's fitness being determined entirely by its ability to survive, interact with other organisms (Fortuna et al. 2013) collect needed resources (Walker & Ofria 2012), and reproduce. Avida is an instance of real evolution occurring in a virtual world (Pennock 2007). Avida has been successfully used in many experiments involving antagonistic coevolution (e.g., Zaman et al. 2011; Fortuna et al. 2013) as wall as altruism (e.g., Goings et al. 2004; Clune et al. 2011) and cooperation (e.g., Knoester et al. 2008; Goldsby et al. 2012, 2014)

Avida organisms consist of a sequence of genetic instructions. An organism must execute genomic instructions to take individual actions, with the dynamic series of actions collectively describing the organism's traits and behaviors (i.e. its phenotype). Potential actions include those governing the sensing and processing of information (e.g., sensory

information about other organisms or objects), movement, and reproduction, as well as instructions controlling internal logic and the order in which genetic instructions are executed. During reproduction, a parent's genome is copied into its offspring, with experimenter-defined mutation probabilities for substitutions, deletions, and insertions. If a substitution or insertion mutation does occur, a new instruction, randomly selected from the full set of all available instructions, is placed into the offspring genome. As a consequence of mutational changes, organisms frequently have new genotypes and express novel phenotypes, yielding associated intrapopulation variation in fitness. Taken together, these conditions allow Avida to satisfy the necessary conditions for adaptive evolution via natural selection: replication, inheritance, variation, and differential fitness.

Environment

Avida environments are built on a grid-cell base, with organisms having an effective physical size of one cell. Thus, all organisms have potential neighbors on eight sides. Here, unlike in most prior Avida experiments, multiple organisms can simultaneously occupy any given cell so collisions between organisms do not need to be considered. (Wagner et al. 2013). All experiments were conducted in a 101 by 101 grid-cell toroidal environment. The toroid design allows for 'borderless' environments, eliminating any artifacts due to boundary or related effects. An additional difference from previous predator-prey Avida experiments (Lehmann et al 2013, Wagner et al. 2013, Fish et al. 2014) is that resources in the environments used here were unlimited and ubiquitous.

Reproduction

In these experiments, organisms were required to eat at least one unit of resource before they were allowed to replicate. Unlike in most other Avida experiments where organisms must individually copy each genomic instruction, we allowed them to execute a single genomic "repro" instruction. Meeting these requirements for reproduction could be trivial, and thus we also set a minimum age for reproduction of 5,000 instructions executed, or approximately 167 updates. Updates are the primary measure of time in Avida, with each organism, on average, executing 30 instructions per update. Organisms can evolve to use up to four execution threads (Ofria et al. 1999), allowing them to process multiple portions of their genome at once. In simple terms, evolving to use multiple threads allows for near-simultaneous execution of multiple actions (akin to walking while chewing gum).

In effect the primary consequence of having a minimum age, with no other major requirements for reproduction, is to ensure that the evolutionary pressure to which prey populations are reacting is simply survival in the face of predation threats. Upon reproduction, there was a 25% chance of a single genomic instruction substitution occurring in the offspring, with 5% chances each for single insertion and deletion mutations. Organisms were born into the cell faced by their parent. Organisms were removed if they did not reproduce before reaching the maximum age limit of 6,000 cycles (≈200 updates).

Predation

The basic mechanics of the predator-prey system in Avida have been described in (Fortuna et al 2013, Lehmannn et al 2013, Wagner et al 2013, Fish et al 2014). In brief, all organisms are born as 'juveniles' that have not expressed any of the behaviors that facilitate resource consumption or predation. In order to consume resources, and be classified as prey, juveniles must execute a 'set-forage-target' instruction. Likewise, any organism that attacks a prey organism (or a juvenile) are automatically classified as a predator. As an alternative to these mechanisms, juveniles can choose to adopt the same predator/prey status as their parent if the parent had executed a 'teach-offspring' instruction and the juvenile executed a 'learn-parent' instruction. In order to promote the early adoption of a prey or predator classification, juveniles in these environments were killed if they attempted to move.

Each organism has a position on the world grid and can face any of the 8 cardinal or diagonal directions. Predation occurs when an organism executes an 'attack' instruction while there is a prey or juvenile organisms in the cell they are facing. If there are multiple non-predator organisms in the faced cell, a random organism is targeted. If a valid target organism does occupy the faced cell and a kill is made, the predator gains all of the resources accumulated by that prey, including those in the form of stored resources (typically in the form of 'bonus'), completed tasks, and merit (see below). Via that intake of completed prey tasks, predators satisfied the 'eat once' prerequisite for reproduction if prey they consumed had fed from the environment. Classified predators were prevented from resetting their forage targets to become prey.

Movement

Because the resources in these experimental environments were always available, prey were under no pressure to move in the absence of predators. However, since predators must evolve out of the ancestral prey population, either prey need to be mobile in order for predators to evolve into the system, or predators must move immediately from the point where they first evolve. Accordingly, we rewarded prey for up to 10 executions of the move instruction by applying small increases in how quickly they could execute their genomic program. In Avida, an organism's merit determines the rate it can execute instructions. That is, an organism with a merit of two will execute twice as many instructions per update as an organism with a merit of one (all organisms are born with a merit of one). Thus prey could increase their metabolic rate simply by moving up to 10 steps, adding one to their merit with each step. This simple mechanism introduced pressures for prey to realize the minimum initial movement rates to facilitate the evolution of predators from the populations.

Populations

Because resources did not limit the number of prey in an experiment, we set a cap of 1000 organisms for each evolving population in every treatment (see below). When the birth of a new organism would have caused the population size to exceed this level, a random organism was removed from the existing population. Additionally, we set a minimum prey population size of 500, below which predation was prevented. In effect, hitting this minimum level serves only to force

predators to retry an attack as reproduction in the system is continuous and so another prey was likely born elsewhere in the system almost immediately following a prevented attack. To standardize levels of predation pressure across trials, predator populations were limited to 50 or 100 individuals (as indicated in the results). If classification of a new predator would push the predator population above the set limit, a randomly selected predator was removed from the population.

Sensors

Evolved strategies can make use of visual sensors capable of providing information about the external environment (i.e. they can evolve sight). The sensory apparatus in Avida is designed to be highly configurable, permitting the evolution of complex sensory strategies. Overall, the sensory system includes four controlling inputs and eight information outputs. Outputs place descriptive integers into particular registers and organisms can determine which registers receive which types of information. Mechanisms allowing the organisms to process and react to that information must also be evolved. Inputs allow organisms to choose to collect information about particular types of objects (e.g., predators vs. prey) over a specified distance. Additionally, organisms can use their sensors to collect more information about a specific object (e.g., the closest of the specified type or one at a specified distance), or a count of all objects of that type in their visual field. Sensory outputs include seen object counts, object values (e.g., value of an environmental resource, or 'fatness' of a prev in terms of their collected resource level), the identity of the closest object of the type searched for in the visual field, the type of object being described (e.g., predator vs. prey), and any group identifiers for the first organism seen (see 'Opinion Confusion', below). Sensors return data from a 45-degree visual field. In these experiments, maximum sight distance was set to 10 cells. A complete list of sensor default behaviors is available in the Avida documentation. Note that ancestral organisms have no sensors. It is entirely up to evolution to discover pathways for vision, a discovery that will not be realized unless vision is a useful adaptation.

Treatments

In order to evaluate the effects of predation pressures on the evolution of prey grouping behaviors, we evolved prey under eight different sets of conditions for one million updates, realizing ≈5,800 prey generations of evolution in each. We initialized all 30 replicates of each treatment by introducing a single, simple prey organism that repeatedly fed, turned randomly, moved, and attempted to reproduce. Treatment names indicated with italics below correspond with their labels in Figure plots.

No Predators (control): In the primary control, prey evolved in environments in which we prevented predators from evolving into the system by preventing the 'attack' instruction from functioning.

Predators: In our baseline predator treatment, predators and prey coevolved without any induced experimenter interference. This treatment offers the primary test of whether predators cause prey to evolve grouping strategies.

No Vision: In this treatment, the visual sensors for all organisms were prevented from functioning. Thus, this treatment, when compared with the *Predators* treatment, offered an indication of the importance of visual information in predator decision-making. That is, if prey form groups even when predators cannot see them, prey are clearly not grouping in order to visually confuse them. Instead, prey grouping here would indicate the dilution effect alone was responsible for prey grouping.

Artificial Confusion: In two treatments we established artificially enforced 'confusion' effects in order to allow direct comparison with previous studies (e.g., Olson et al. 2013). We do not contend that either of these treatments actually induces confusion (in contrast with Olson et al. 2013). Rather, they attempt to mimic the hypothesized outcomes of an attack if predators are confused. Specifically, in the Artificial Confusion treatment, the odds of a predator successfully making a kill, given an attack, is set to 1.0 -(0.10 * r), where r is the number of prey neighboring the targeted prey. Thus a prey would be completely protected if there were 10 other prey in its cell or in neighboring cells. Similarly, in the Artificial Confusion Inverse treatment, the odds of a predator making a successful kill were set to 1/r. These odds operate independent of a predator's true understanding of visual cue information. Thus, it enforces a hypothesized outcome of confusion, even in the absence of actual predator confusion.

Visual confusion: In the three remaining treatments, we created conditions that would allow prey to evolve behaviors that would, explicitly, confuse predators by reducing the accuracy of sensory information. Specifically, each of these three treatments introduces a probability that the predator will receive random data from visual sensors when looking at prey, with the probability being dependent on the behavior of the prey seen. If triggered, the visual outputs for determining the distance to the first prey in sight, the counts of prey in the visual field, the value ('fatness') of the first prey, and the group id ('opinion', see below) of the first prey would all be random numbers. In effect, these treatments offer opportunities for prey to evolve behaviors that 'blur' the vision of their attackers.

In the *Density Confusion* treatment, the odds of predators receiving random visual inputs was, as in the *Artificial Confusion* treatment, set to 1.0 - (0.10 * r). Unlike the *Artificial Confusion* treatment, however, this equation did not directly define the odds of an attack succeeding or failing.

Similarly, in the *Opinion Confusion* treatment, the odds of predator receiving confused visual information was set to 1.0 - (e/o), where e is the number of opinions being expressed by the nearest prey and its neighbors and o is the total number of opinions possible. Opinions effectively allow prey to form group identities, outside of the predator-prey classification system. Here, there were four possible opinions (0,1,2,3). Accordingly, a predator will always receive confused visual information whenever four opinions are being simultaneously expressed in the group of prey around the prey nearest the predator. Opinions are set whenever an organism executes a

'join-group' instruction, with the modulus of the value in the associated register determining the opinion used.

Finally, we included a treatment in which the physical orientation of prey determined the odds of a predator's vision being confused. Specifically, in the *Facing Confusion* treatment, the odds of a predator confused visual information was set to 1.0 - (f/8), where f is the number of different directions the prey (those surrounding the prey nearest the predator) are facing.

Measuring Grouping

In order to quantify the realized levels of prey clumping, we used a density-based measure that considered the position of all organisms in the two-dimensional grid environment. Since multiple organisms could occupy any cell, we calculated prey clumping as 1 - (n/p), where n is the number of unique cells occupied and p is the total number of prey organisms in the system. Thus, a clumping value approaching 1.0 indicates that all prey were in the same single cell (a maximum grouping scenario), a value of 0.5 would indicate that prey were clumped with an average of two individuals per cell, and a value of zero would indicate little to no clumping (each organism in their own cell).

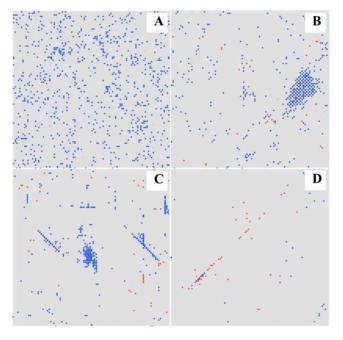


Figure 1: The dilution effect alone drives the evolution of prey grouping strategies. Prey (blue points) evolved in the absence of predators (A) show no tendencies toward spatial grouping. However, when coevolving with predators (red; B-D), the dilution effect quickly drives prey to form large and tight groups for protection, even in the absence of any confusion effects. Fewer 'visible' prey indicates that they are sharing cells (overall population sizes are consistent across images).

Software

We used Avida version 2.13 for all experiments, using the EX virtual hardware as in Wagner et al 2013 and Lehmann et al. 2013. Among other modifications to the base Avida system, the EX virtual hardware sets the definitions and capacities for organisms to evolve predation and visual sensors. Data were post-processed using Python 2.7.1. Statistical analyses and plotting were conducted in R version 2.15.2 using the ggplot2 library.

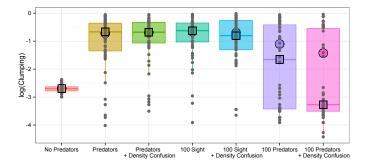


Figure 2. Prey evolve tight aggregation strategies, regardless of predator confusion. Shown are the mean (large points) and median (small boxes) clumping densities of prey under each predator treatment. Values for individual populations are shown with small grey points. Even when predators were prevented from evolving vision (i.e. visual confusion not possible), prey evolved to live in groups as a consequence of the dilution effect. Thus confusion cannot drive the evolution of prey grouping behaviors.

Results

After 1 million updates of evolution, even in the absence of any confusion effects, the presence of predators pushes prey to evolve aggregation strategies (Figure 1, online video at http://youtu.be/J8dQYroyA88), with the measured level of prey clumping increasing from a mean of 0.068 (min = 0.05, max = 0.093, sd = 0.009) with *No Predators* to 0.485 (min = 0.017, max = 0.948, sd = 0.273) with *Predators* (Figure 2).

When predators' odds of making a kill are artificially reduced based on local prey densities (i.e. *Artificial Confusion* and *Artificial Confusion Inverse*), relatively to the default *Predators* treatment, clumping levels cover a narrower breadth of values (mean = 0.402, min = 0.103, max = 0.934, sd = 0.193 for *Artificial Confusion*, mean = 0.309, min = 0.0704, max = 0.844, sd = 0.172 for *Artificial Confusion Inverse*). However, even these imposed reductions in attack success rates do not significantly increase grouping; prey clumping values remain in the range of the majority of the simpler, and far more natural, *Predator* treatments. Similarly, *Density, Opinion* (prey type), and *Facing* confusion mechanisms did not substantially impact the extent of clumping (Density: mean = 0.500, min = 0.030, max = 0.951, sd = 0.254; *Opinion* mean = 0.549, min = 0.042, max = 0.956,

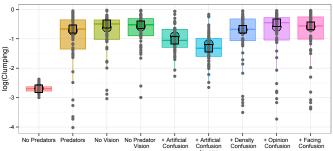


Figure 3. Sight distance limitations do not impact the evolution of aggregation strategies. Shown are the mean (large points) and median (small boxes) clumping densities of prey under additional treatments that extended the maximum sight distance of organisms out to 100 cells (the width and height of the world) and, seperately, treatments that raised the predator population cap from 50 to 100. Values for individual populations are shown with small grey points. No Predators, Predators, and Density Confusion data are repeated here (from Figure 2) for comparison.

sd = 0.273; Facing mean = 0.550, min = 0.034, max = 0.956, sd = 0.281). Additionally, extending the maximum sight distance to 100 also had no appreciable effect over that introduced simply by the introduction of predation pressures (Figure 4). At the same time, increasing the maximum predator population size to 100 greatly extended the spread of the data, with fewer prey populations evolving to form tight groups (Figure 4). We speculate that the latter effect is related to reductions in the protection provided by the dilution effect when predator attack rates are very high (Figure 5).

The lack of difference between the *Predators* and *No Predator Vision* treatments further highlights the strength of

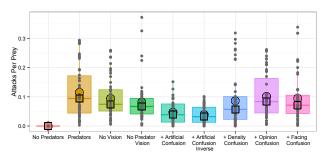


Figure 4. Prey behaviors do not decrease experienced attack rates. Shown are the mean (large points) and median (small boxes) attacks per prey (i.e. kill rate) under each predator treatment. Values for individual populations are small grey points. Because prey clump simply in response to predators, thereby already minimizing their vulnerability to attack, only imposed confusion treatments (which directly control attack success probabilities, not confusion), have noticable effects on attack rates. Attacks per prey was calculated as the total lifetime number of kills for all predators, divided by the prey population size.

the dilution effect over any confusion effects. That is, even though predators can clearly evolve effective methods for using visual sensors in hunting (e.g., Figure 6, online video at http://youtu.be/HTWzWcm7jMs), the dilution effect continues to push prey to clump even when predators are blind. Clearly, when predators are blind, confusion plays no role.

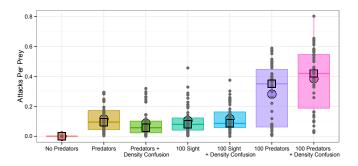


Figure 5. Maximum attack rates scale with predator densities, but not sight distance. Shown are the mean (large points) and median (small boxes) attack rates under treatments that extended the maximum sight distance of organisms out to 100 cells (the width and height of the world) and, seperately, treatments that raised the predator population cap from 50 to 100. Values for individual populations are shown with small grey points. No Predators, Predators, and Density Confusion data are repeated here (from Fig. 3) for comparison. In general, very high attack rates may dissolve any benefits accrued via the dilution effect, thereby reducing the tendency of prey to aggregate.

Overall, these results indicate that the presence of predators alone induces prey clumping, primarily due to the dilution effect. Furthermore because they are already clumped, any confusion processes have no impact on the tendency to form aggregations.

Effects of the treatments on attack rates experienced by prey (Figure 3) tell a story similar to that of clumping behavior: dilution effects dominate confusion effects and once clumped, prey populations do not further substantially reduce attack rates via the evolution of any confusion behaviors. Mean (min-max; sd) attack rates for each treatment were *Predators* = 0.115 (0.003 - 0.294; 0.089), *No Vision* = 0.0942 (0.006 - 0.259; 0.0673), *No Predator Vision* = 0.0812 (0.003 - 0.372; 0.071), *Artificial Confusion* = 0.0474 (0.0 - 0.152; 0.038), *Artificial Confusion Inverse* = 0.0395 (0.0 - 0.101; 0.030), *Density Confusion* = 0.0866 (0.0 - 0.319; 0.880), *Opinion Confusion* = 0.101 (0.005 - 0.262; 0.0787), and *Facing Confusion* = 0.0948 (0.002 - 0.339; 0.0819).

Discussion

We evaluated the factors contributing to the natural evolution of grouping in prey species coevolving with predators. Specifically, we explored differences in evolved tendencies of prey to form groups in order to gain protection from predators as a consequence of simple dilution, naturally evolved confusion behaviors, and artificially induced confusion effects. Our results clearly show that grouping tendencies increase with the introduction of predators, regardless of any confusion effect or behavioral treatments (Figure 2). This increased tendency to aggregate appears to be due entirely to the inherent dilution effect.

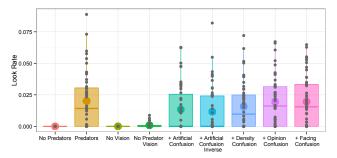


Figure 6. When permitted, predators evolve vision for hunting in most populations. Shown are the evolved mean (large points) usage of visual sensors by predators as a proportion of their total lifetime instruction exeuctions. Values for individual populations are shown with small grey points. When attack success is artificially reduced (Artificial Confusion treatments), median rates of sensor use by predators sits near zero. Otherwise, predators evolve to use sight even though it is not a requirement for successful predation. Values above zero for No Predator Vision treatments indicate predators that used their sensors while still in a juvenile state, presumable due to drift.

Additionally, it is useful to realize that the root evolutionary cause of prey aggregations is not a cooperative strategy. Instead, as it is ultimately a consequence of the dilution effect, the initial selective pressures for grouping arise from selfish movement strategies for minimizing ones own probability of being subject to an attack. While this self-interest will drive prey to aggregate, the resulting 'swarms' simply represent groups of organisms exhibiting correlated, selfish strategies, not cooperative behaviors.

The limited effects of blinding predators and of increasing potential sight distance highlights additional reasons why confusion behaviors are of only marginal importance in the evolution of swarming. After all, if predators cannot see prey, prey cannot visually confuse them. Moreover, because predators and prey coevolve, arms race dynamics dictate that predators will evolve behavioral counter-measures to prey anti-predator behaviors. Accordingly, attack rates are expected to remain consistent across treatments, as they do here (Figure 3). More broadly, in order for there to be a confusion effect, predators need to evolve visual perception. The dilution effect however, is simply achieved whenever prey aggregate, whether intentionally or by chance, thereby "diluting" any one prey's odds of being targeted, without diminishing the overall chances of a predator making a kill. The dilution effect requires fewer conditions and is, on the whole, a simpler and more fundamental effect (Foster 1981, Delm 1990).

Although they do not promote the evolution of grouping behaviors, various forms of confusion may still have subtle effects on prey behaviors. Specifically, while dilution directly controls the probability of being selected and attacked by a predator, confusion behaviors can reduce the probability of being killed given an attack. Thus, it is the dilution effect that drives the evolution of anti-predator prey grouping strategies. Once prey are grouping, however, the further evolution of confusion behaviors can continue to benefit prey, even if they are not responsible for the evolutionary origin of grouping per se. In other words, dilution causes prey to evolve swarming, while confusion behaviors can provide additional benefits once grouped.

Acknowledgments

We thank G. Wright for his assistance in developing components of the experimental system and members of the MSU Digital Evolution Laboratory for fruitful discussion. This work was supported by the BEACON Center for the Study of Evolution in Action (NSF Cooperative Agreement DBI-0939454) and the Michigan State University Institute for Cyber Enabled Research.

References

Abrams, P. A. (2000). The evolution of predator-prey interactions: theory and evidence. *Ann. Rev. of Ecology and Systematics*: 79-105.

Beekman, M., and F. L. W. Ratnieks. (2000). Long-range foraging by the honey-bee, Apis mellifera L. *Functional Ecology* 14.4 490-496.

Bengtson, S. Origins and early evolution of predation. *Paleontological Society Papers* 8 (2002): 289-318.

Boreham, M. M., and David W. R. (1987). Population change and control of Africanized honey bees (Hymenoptera: Apidae) in the Panama Canal area. *Bulletin of the ESA* 33.1 34-39.

Bryson, D.M. & Ofria, C., (2013). Understanding Evolutionary Potential in Virtual CPU Instruction Set Architectures. *PLoS One* 10.1371/journal.pone.0083242.

Clune, J., Goldsby, H. J., Ofria, C., & Pennock, R. T. (2011). Selective pressures for accurate altruism targeting: evidence from digital evolution for difficult-to-test aspects of inclusive fitness theory. *Proceedings of the Royal Society B: Biological Sciences*, 278(1706), 666-674.

Cosner, C., et al. (1999). Effects of spatial grouping on the functional response of predators. *Theoretical Population Biology* 56.1: 65-75.

Dawkins, R. and John R. K. (1979). Arms races between and within species. *Proceedings of the Royal Society of London. Series B. Biological Sciences* 205.1161: 489-511.

Delm, M. M. (1990). Vigilance for predators: detection and dilution effects. *Behavioral Ecology and Sociobiology* 26.5 337-342.

Fish, J., O'Donnell, D. R., Parigi, A., Dworkin, I., & Wagner, A. P. (2014). The roles of standing genetic variation and evolutionary history in determining the evolvability of anti-predator strategies. *bioRxiv*.

Fortuna, M. A., Zaman, L., Wagner, A. P., & Ofria, C. (2013). Evolving digital ecological networks. *PLoS computational biology*, *9*(3), e1002928.

Foster W.A. and Treherne J.E. (1981) Evidence for the dilution effect in the selfish herd from fish predation on a marine insect. 466-467.

Goings, S., Clune, J., Ofria, C., & Pennock, R. T. (2004, September). Kin selection: The rise and fall of kin-cheaters. In *Proceedings of the Ninth International Conference on Artificial Life* (pp. 303-308).

Goldsby, H. J., Dornhaus, A., Kerr, B., & Ofria, C. (2012). Task-switching costs promote the evolution of division of labor and shifts in individuality. *Proceedings of the National Academy of Sciences*, 109(34), 13686-13691.

Goldsby, H. J., Knoester, D. B., Ofria, C., & Kerr, B. (2014). The Evolutionary Origin of Somatic Cells under the Dirty Work Hypothesis. *PLoS biology*, *12*(5), e1001858.

Gueron, S. and Simon A. L. (1993). Self-organization of front patterns in large wildebeest herds. *Journal of Theoretical Biology* 165.4 541-552.

Jakobsen, P. J., and Geir H. J. (1988). Size-specific protection against predation by fish in swarming waterfleas *Bosmina longispina*. *Animal behaviour* 36.4 986-990.

Jeschke, J. M., and Tollrian R. (2007). Prey swarming: which predators become confused and why? *Animal Behaviour* 74.3 387-393

Karanth, K. U., and Melvin E. S. (2000). Behavioural correlates of predation by tiger (*Panthera tigris*), leopard (*Panthera pardus*) and dhole (*Cuon alpinus*) in Nagarahole, India. *Journal of Zoology* 250.2, 255-265.

Knoester, D. B., McKinley, P. K., & Ofria, C. (2008). Cooperative network construction using digital germlines. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation* (pp. 217-224). ACM.

Krakauer, D. C. (1995). Groups confuse predators by exploiting perceptual bottlenecks: a connectionist model of the confusion effect. *Behavioral Ecology and Sociobiology* 36.6 421-429.

Krams, I., et al. (2010). The increased risk of predation enhances cooperation. *Proceedings of the Royal Society B: Biological Sciences* 277.1681 513-518.

Kunz, H., T. Züblin, and C. K. (2006). Hemelrijk. On prey grouping and predator confusion in artificial fish schools. *Proceedings of the Tenth International Conference of Artificial Life. MIT Press, Cambridge, Massachusetts*.

Landeau, L., and John T. (1986). Oddity and the 'confusion effect'in predation. *Animal Behaviour* 34.5 1372-138.

Lehmann K, BW Goldman, I Dworkin, D Bryson & AP Wagner. (2014). From cues to signals: Evolution of interspecific communication via aposematism and mimicry in a predator-prey system. *PLOS ONE*.

Milinski, M. (1984). A predator's costs of overcoming the confusion-effect of swarming prey. *Animal Behaviour* 32.4 1157-1162.

Milinski, M. (1979). Can an experienced predator overcome the confusion of swarming prey more easily?. *Animal Behaviour* 27 1122-1126.

- Mooring, M. S., and Benjamin L. H. (1992). Animal grouping for protection from parasites: selfish herd and encounter-dilution effects. *Behaviour* 173-193.
- Nagy, K. A., and John N. S. (1976). Temperature maintenance and CO2 concentration in a swarm cluster of honey bees, *Apis mellifera*. *Comparative Biochemistry and Physiology Part A: Physiology* 55.2 169-171.
- Ofria, C., Adami, C., Collier, T.C., and Hsu, G.K. (1999). Evolution of Differentiated Expression Patterns in Digital Organisms. *Advances in Artificial Life* 129-138.
- Ofria, C., Bryson, D. M., & Wilke, C. O. (2009). Avida: A Software Platform for Research in Computational Evolutionary Biology. *Artificial Life Models in Software*.
- Okubo, A. (1986). Dynamical aspects of animal grouping: swarms, schools, flocks, and herds. *Advances in biophysics* 22 1-94.
- Olson, R. S., Hintze, A., Dyer, F. C., Knoester, D. B., and Adami, C. (2013). Predator confusion is sufficient to evolve swarming behaviour. *Journal of The Royal Society Interface*, 10(85), 20130305.
- Omholt, S. W. (1987). Thermoregulation in the winter cluster of the honeybee, *Apis Mellifera*. *J. theoretical biology* 128.2 219-231.
- Partridge, B. L. (2007). The structure and function of fish schools. *Scientific American* 246.6 (1982): 114-123.
- Pennock, R.T. (2007). Models, simulations, instantiations, and evidence: the case of digital evolution. *J. Exp. Theor. Art. Int.* **19**, 29-42
- Relyea, R. A. (2001). Morphological and behavioral plasticity of larval anurans in response to different predators. *Ecology* 82.2 523-540.
- Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*. 21(4)
- Salge, C., & Polani, D. (2011). Local information maximisation creates emergent flocking behaviour. *Advances in Artificial Life*, ECAL 2011.
- Silk, J. B. (2007). Social components of fitness in primate groups. *Science* 317.5843 1347-1351.
- Sivinski, J. M., and Petersson E. (1997). 17'Mate choice and species isolation in swarming insects. *The evolution of mating systems in insects and arachnids* 294.
- Turner, G. F., and Pitcher T.J. (1986). Attack abatement: a model for group protection by combined avoidance and dilution. *American Naturalist* 228-240.
- Wagner, A. P., Zaman, L., Dworkin, I., & Ofria, C. (2013). Arms Races and Strategy Chases Promote the Evolution of Prey Intelligence. *arXiv preprint arXiv:1310.1369*.
- Walker, B. L., & Ofria, C. (2012). Evolutionary potential is maximized at intermediate diversity levels. In *Artificial Life* (Vol. 13, pp. 116-120).
- Zaman, L., Devangam, S., & Ofria, C. (2011, July). Rapid host-parasite coevolution drives the production and maintenance of diversity in digital organisms. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation* (pp. 219-226).

The Evolution of Learning Under Environmental Variability

Kai Olav Ellefsen

Department of Computer and Information Science, Norwegian University of Science and Technology email: kaiolae@idi.ntnu.no

Abstract

An important unanswered question within the evolution of intelligence is how evolved learning efforts relate to environmental characteristics. Through simulated evolution of simple learning agents, we study two different models of the relationship between environmental variability and evolved learning. We begin with a recently proposed model (the "pqmodel"), which suggests that 1) unreliable reinforcing feedback select against learning and 2) a fixed environment selects for innate strategies, whereas a changing environment selects for learned strategies. The other model we study proposes that, in contrast with point 2 above, intermediate values of environmental stability select for learning, whereas both too stable and too variable environments will select against it. We harmonize these seemingly conflicting models by evolving learning agents across a wide range of environments, varying in levels of stability and reliability of stimuli. Based on our findings, we propose a revised model for how learning evolves under different levels of environmental variability.

Introduction

In an evolutionary context, we would intuitively expect a learning individual to outperform non-learners in a changing environment. In a stable environment, on the other hand, we would expect individuals with innate behaviors to do better, in particular if learning is associated with a cost (De-Witt et al. (1998); Mery and Kawecki (2002)). Dunlap and Stephens (2009) argue that this view of learning, where a changing environment promotes plasticity while a static environment promotes stability (the "learning folk theorem") is too simplified. Through a mathematical model (from here on referred to as the pq-model) and an evolutionary experiment on fruit flies, they identify two different factors of environmental change that affect the evolution of plasticity in opposite directions. The first type of change, termed bestaction fixity describes to which degree the best action to take in the environment is always the same. The second type of change, termed reliability of experience represents the reliability of reinforcing feedback. According to the pq-model (Figure 1), a higher reliability of experience selects more strongly for learning, while a higher fixity of the best action selects more strongly for non-learning strategies. Intuitively,

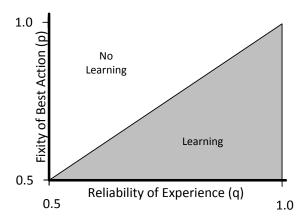


Figure 1: The pq-model (Dunlap and Stephens (2009)) of evolved learning: Environments where the optimal strategy never changes (p=1) select most strongly against learning. Environments where experiences cannot be trusted (q=0.5) do the same. By similar reasoning, (p=0.5, q=1) select most strongly for learning.

we can think of it like this: Stable associations (high fixity of best action) do not require learning – thus selecting against it, and noisy sensors (low reliability of experience) makes learning difficult or impossible – also selecting against it. Details about the model, such as what the actual values of p and q mean will be given in the Experimental Setup section.

Another view of how environmental change affects the learning ability of individuals (Kerr and Feldman (2003); Dukas (1998)) suggests that the relationship between environmental variability and the utility of learning follows the "Goldilocks principle" (Figure 2): For learning to be beneficial, environmental variability needs to "just right" – too frequent changes, and learning cannot track them. Too infrequent, and evolution can track them alone.

There seems to be a conflict between these two models for the relationship between environmental change and plasticity. The type of environmental change studied in the "Goldilocks principle"-model is what Dunlap and Stephens termed "fixity of best action": How often does the best be-

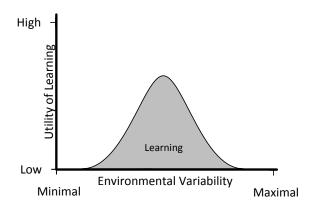


Figure 2: The "Goldilocks principle"-model of evolved learning: Learning evolves in an intermediate range of environmental variability. Too slow or too rapid changes select against learning.

havioral strategy in the environment change? According to the pq-model, low levels of best-action fixity are associated with a high learning ability. The Goldilocks principle, on the other hand, states that both extremes should *disfavor* learning.

We believe the source of the conflict is the way Dunlap and Stephens defined their lowest level of best-action fixity: It is simply not low enough, and rather placed somewhere in *the middle* of the range of possible environmental change – which would make it a candidate for high levels of learning, according to the Goldilocks principle.

The goal of this paper is to study the pq-model through an experiment with simulated organisms evolved under different degrees of evolutionary change, to get a better understanding of how the reliability of experience and fixity of best action influence their learning ability. Next, we will attempt to unify the pq-model with the Goldilocks principle model in a single framework. We will also study how the model changes when we apply the biologically realistic constraint of a *cost* of learning (Mery and Kawecki (2002); Mayley (1996)).

Related Work

The experimental setup of Dunlap and Stephens was based on a preparation developed by Mery and Kawecki (2002). Using this preparation, Mery and Kawecki tested and confirmed the "learning folk theorem" – that changing environments select for learning, while stable environments select against it. Challenging this experimental confirmation was one of the goals of Dunlap and Stephens, and therefore we find it relevant to briefly review Mery and Kawecki's experimental setup and findings.

Mery and Kawecki (2002) wanted to study how learning evolves under natural conditions, where the costs of learning will have to be outweighed by its benefits, for learning to evolve. For this reason, they developed an experiment testing under which circumstances learning would evolve in a population of fruit flies. The experiment had two different phases: In the first phase (training), the flies were subjected to two fruit-flavored media, one of which (alternating each generation) additionally contained quinine hydrochloride. Flies showed a strong avoidance to the medium paired with quinine. In the next phase (testing), the quinine cue was not present, and flies were again subjected to the media, this time laying eggs. Flies which had learned a strong association between quinine and the fruit flavored medium would lay most of their eggs on the opposite medium. When breeding the next generation of flies, the eggs from this opposite medium were selected, giving a pressure for flies to learn in order to succeed in reproducing.

The results showed that after several generations of evolution, the presence of quinine in the training phase affected the choice of oviposition substrate (where to lay eggs) in the testing phase, an effect which grew stronger as generations passed – demonstrating that environmental characteristics can affect evolved learning abilities.

Environmental change and plasticity

Computational experiments are well-suited to study the relationship between environmental change and evolved plasticity, as the environmental change can be tuned exactly as needed, and as generations can be completed very rapidly compared to in animal studies. These facts allow studies spanning many types of environments and a large number of generations. Sasaki and Tokoro (1999) studied how rates of change in an environment affected populations of evolving individuals, finding that the benefit of learning agents (as opposed to purely genetically specified agents) was greater in the more dynamic environments than in more stable ones.

Our previous experiments (Ellefsen (2013)) have suggested a similar trend: Allowing evolution to shape the degree of learning vs genetic specification, we saw hard-coded individuals evolve in more stable environments, and learning agents evolve in the less stable ones. Further, we discovered that the *most unstable* environments had a tendency to select against learning, a finding also reported in other experiments on the evolution of plasticity (Watson and Wiles (2002)). Together, these computational studies of the relationship between rates of environmental change and evolved learning abilities lend support to the Goldilocks principle model of learning: Too rapidly or too slowly changing environments were both seen to select for hard-coded strategies, while learning was evolved in environments with intermediate levels of change.

Learning Costs

The benefits of learning are frequently documented in studies of interactions between evolution and learning. Many experiments (see for instance Floreano and Urzelai (2001);

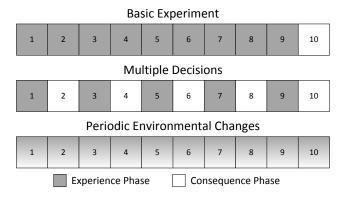


Figure 3: Illustrations of the different phases in the three different experimental setups we use. See text for details.

Littman (1995); Nolfi and Parisi (1996); Nolfi et al. (1994)) have highlighted how learning, as a supplement to evolution, gives agents the ability to respond to rapidly changing conditions, to adjust to different environments and morphologies, and to solve more general problems than evolution could alone.

When studying the evolution of learning, it is important to also remember that learning has a *cost* (DeWitt et al. (1998); Mayley (1996)). It is the balance between the cost and benefit of learning that decides the final learning strategies followed by individuals resulting from an evolutionary process. An implication of the cost of plasticity is that plasticity in organisms needs to have *adaptive value*. When possible, natural selection will reduce costs by replacing plastic responses with genetic mechanisms. This will have an influence on the relationship between environmental change and evolved learning efforts. Therefore, we study our models under three conditions: without an externally imposed plasticity cost and with two different levels of this cost.

Experimental setup

All experiments were performed with a discrete-generation evolutionary algorithm where all individuals had the same lifespan and fitness was based on the decision(s) made throughout life. Within this general setup we performed different experimental treatments, with increasing levels of sophistication and realism, as explained below.

The basic experiment

The first experiment simulates Dunlap and Stephens' original setup. Their experimental setup consisted of two phases, the *experience phase* and the *consequence phase*. In the experience phase, fruit flies were exposed to two fruit flavors, pineapple and orange, for three hours. One of the flavors was associated with quinine, enabling flies to learn to avoid one of the fruits in this phase. In the consequence phase, both fruit flavored media were presented without quinine. In this phase, flies laid eggs, leading to more eggs from the best

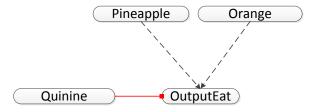


Figure 4: A neural network learning the association from food items to edibleness via quinine association. Rounded rectangles are neurons, the dashed lines are learning links that are modified by Hebbian learning and the solid line is a static, negative connection simulating the fruit fly's negative response to the quinine (Mery and Kawecki (2002)). Inputs are binary, representing the presence or absence of the relevant stimulus. The output at any time gives a preference score for the current fruit, and the highest scoring of the two is named the preferred (fitness-deciding) fruit.

learners landing on the medium that did not contain quinine in the experience phase. To vary the environmental stability, Dunlap and Stephens varied 1) the probability that the eggs laid on the orange were selected for reproduction (p) and 2) the probability that the eggs laid on the medium not associated with quinine were selected for reproduction. The first allows regulation of environmental stability, and the second allows variation of how predictive the quinine association is of reproductive success.

In our replication of this experiment, we split the task in an experience and consequence phase in a similar fashion. In the experience phase, fruit flies *observe* both fruits 49 times, one of the two giving a quinine-stimulus in addition to the fruit flavor. In the consequence phase, both fruits are presented again without quinine, and the fly computes a preference score for each fruit. The most preferred fruit is visited (or, equivalently, eaten), and this *single visit* decides the fitness score (100 for the "good" fruit, -100 for the "bad" fruit). Note that there is a certain simplification here: The fly only makes a single decision, which equals laying all its eggs on one medium. However, in the experiment on real fruit flies, each fly laid multiple eggs. We decided to make this simplification, as the two extensions we developed for the task specifically deal with the option of multiple decisions.

The top part of Figure 3 illustrates this experiment. Note that this figure is not to scale: In the actual experiment, the consequence phase was even shorter compared to the experience phase (1 decision vs. 49 observations). The same neural network (Figure 4) was used in this and the next experiment. The network can learn the association from food item to edibleness when the quinine is present via Hebbian learning (Hebb (1949)), and the learned association is used when classifying food items *without* the quinine present.

Extending to multiple decisions

In Dunlap and Stephens' setup, a single learning phase followed by a single decision phase forms the basis for the reproductive chances of each individual. In a more realistic scenario, survival and reproductive success will depend on learning experiences and decisions made throughout life. This is simulated by letting each individual go through 25 minimal experience phases followed by consequence phases in a lifetime. In this case, experience phases consist of a single observation of each fruit (paired with quinine when appropriate), and the consequence phase is identical, but without the quinine. Individuals receive fitness points for each consequence phase (+1 for the "good" fruit, -1 for the "bad" fruit), and the total fitness equals the sum of fitness points gained in all the consequence phases.

Notice that this setup (Figure 3, middle) also deviates from natural learning processes in that it sharply separates learning-periods and fitness-determining periods.

Extending to periodic environmental changes

The final and most realistic model has been used it previous studies of the evolution of associative learning (Todd and Miller (1991); Ellefsen (2013)). This model views learning and fitness-evaluation as overlapping, continuously occurring processes throughout individuals' lifetimes. The model contains enough detail to allow us to harmonize the two theories we focus on in this paper.

To move away from looking at learning and fitness measurement as two processes happening at different times, we need to slightly alter the decision task. We cannot rely on the quinine as the cue for associative learning, because the quinine naturally divides the learning task into separate training and testing phases. Instead, we look to a similar experiment, where associations are continuously tested and trained by reinforcing stimuli: Agents are presented with one of the two edible substances, and decide if they want to eat it or not. If they decide to eat it, they get a reinforcing feedback in the next time step indicating whether the food was poisonous or edible. This way, there is no separate training and test phases: In each time step, agents receive a new food item and any reinforcing feedback from their previous decision. Agents receive a fitness point whenever eating the edible substance, and lose a point whenever eating the poisonous one.

In the original experiment, the best action fixity (p) was regulated to make the best action change at most once per generation. In our experiment, this is not enough: That change rate will not allow us to see the full range of evolved plasticity levels necessary to observe the Goldilocks principle. Instead, we make the environment *change periodically* – meaning we can regulate the rate of change freely, by adjusting the average length of stable periods.

This setup is illustrated at the bottom of Figure 3. Compared to Dunlap and Stephens' original setup, it differs in

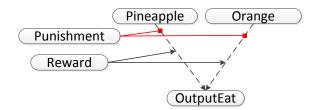


Figure 5: A neural network learning the association from food items to edibleness with continuous reinforcement. Rounded rectangles represent neurons. The dashed lines are learning links that are modified by neuromodulated Hebbian learning. The solid lines are neuromodulatory connections. Inputs are binary, representing the presence or absence of the relevant stimulus. The output at any time gives a preference score for the current fruit. Any fruit preferred above a threshold of 0.5 is "eaten".

two important ways: 1) There is no separate experience and consequence phase. Instead individuals constantly adapt to environmental changes, and are fitness evaluated throughout life. 2) Environmental changes are periodic, allowing us to regulate their rate between the two extremes of no stability and full environmental stability.

The network used to learn this task (Figure 5) uses *neu-romodulated reinforcement* (Soltoggio et al. (2007); Ellefsen (2013)) to form preferences. The neuromodulatory input changes the learning rate temporarily for the links it targets, allowing efficient reinforcement learning.

Parameters

The experimental parameters from Dunlap and Stephens' experiment, fixity of best action (p) and reliability of experience (q) are also the most important parameters here. In the different experimental setups we employ, p has slightly different meanings as a natural consequence of the differences between the setups. These differences are explained here. a in all cases has the same meaning as in the original experiment by Dunlap and Stephens: "The chance that the reinforcing signal (quinine in Figure 4, reward and punishment in Figure 5) is correct.". In other words, this signals to which degree the individual can successfully learn the task (and thus get a high fitness score) by following the reinforcing feedback. This parameter ranges from 0.5 to 1, the lowest value meaning the signal is correct 50% of the time (making it useless for this binary decision task), and 1 meaning it is correct 100% of the time.

In the *basic setup*, p equals the chance that the fruit yielding the high fitness *will be the orange*. It varies from 0.5 to 1, as the values from 0 to 0.5 would be symmetrical. This is identical to its role in the original experiment by Dunlap and Stephens. High values of p indicate a stable environment,

where genetically encoded strategies may be most beneficial.

In the *multiple-decision setup*, *p* still signals the chance that the fruit yielding the high fitness *will be the orange*. However, since we have multiple experience and consequence phases here, this value (as well as q) is tested 25 times per individual instead of once.

In the *periodically changing* setup, p signals the *stability period* of the environment measured in generations. A value of 1 means the environment changes once each generation. This is similar to a p-value of 0.5 in the basic setup, but not identical as we are now dealing with periodic instead of random changes. A value above 1 means the environment changes more seldom than each generation. This is similar to a p-value between 0.5 and 1 in the basic setup, values closer to 1 corresponding to a longer period of environmental stability. Stability periods below 1 indicate several environmental changes per generation. For instance, p = 0.25 means we have 4 changes each generation. The situation with changes within a generation was not considered in the original setup by Dunlap and Stephens.

Neural network learning

As discussed above, we utilize two neural network models in this work, one employing purely *Hebbian learning*, and the other *neuromodulated learning*. The former (Figure 4) utilizes a simple rule that increases the connection strength between co-active neurons:

$$\Delta w_{ij} = \eta * x_i x_j \tag{1}$$

where η is the evolved learning rate and x_ix_j is the product of pre-synaptic and post-synaptic activity. This weakens the relevant connection if the quinine is active (it supplies an inhibitory input, making x_j negative), and strengthens it otherwise. This produces similar associative learning dynamics as in the experiment by Dunlap and Stephens.

The evolved neuromodulated links (Figure 5) update their weights by the following equation:

$$\Delta w_{ij} = \eta * mod * |x_i x_j| \tag{2}$$

This weight is updated by the *absolute value* of a Hebbian term multiplied by the modulatory signal, *mod*. Thereby, the modulation can regulate the direction of the weight change, leading active links to weaken when eating punishers and strengthen when eating rewarding food items.

Evolved variables

In all three experimental setups, three parameters of the neural network were evolved: the innate weight vector (\vec{w}) , the learning rate (η) and the weight decay per time step (λ) . \vec{w} codes for the two weights from Pineapple and Orange to OutputEat. Evolving \vec{w} allows evolution to regulate *innate* behaviors, thus removing the need to learn in static or

Parameter	Value
Generations	50
Adults	50
Children	50
Crossover probability	0.01
Mutation probability	0.005
Genes per individual	4
Bits per gene	8
Elite fraction	0.1
Culling fraction	0.1

Table 1: Parameters of the Evolutionary Algorithm

very slowly changing environments. η is the parameter we are mainly interested in measuring – it tells us which learning efforts are evolved for each environmental setup. Finally, λ is present for the experiments where the optimal behavior changes within individual lifetimes, to allow regulation of how rapidly behaviors are forgotten. Regulating λ and η together lets evolution tune the forgetting and re-learning of behaviors to find an optimal level of plasticity. All parameters were encoded and evolved as 8-bit genes translated into floating-point values for fitness evaluation. The ranges for the evolved variables were: weights [0,1], η [-1,1] (negative learning rates were rounded up to zero, allowing evolution to easily remove learning completely) and λ [0,0.5].

Evolutionary setup

Table 1 gives the parameters of the applied evolutionary algorithm. Crossover probability gives the probability of crossover *per individual*, and mutation probability gives the probability of mutation *per bit* in the individuals' genotype. The values coded for by the four genes are \vec{w} (2 genes), η and λ . See the previous section for their meaning.

Results

The basic experiment

The first thing we wanted to investigate, was whether Dunlap and Stephens' pq-model for learning was correct for their suggested fruit fly experiment. Dunlap and Stephens confirmed their model through experiments on real fruit flies for two extremes of the model: (p=0.5,q=1.0) and (p=1,q=0.5). We wanted to investigate evolved learning efforts also for the rest of the possible p- and q-values.

To do this, we systematically varied *p*- and *q*-values, and evolved agents for 50 generations. 20 runs were done for each parameter combination, and the *average learning rate* of all agents in the final generation was recorded (Figure 6).

Dunlap and Stephens' model proposed that the border between the learning region and the non-learning region would be at p=q, and the evolved individuals lend support to their hypothesis (Figure 6a). Their model makes no prediction for

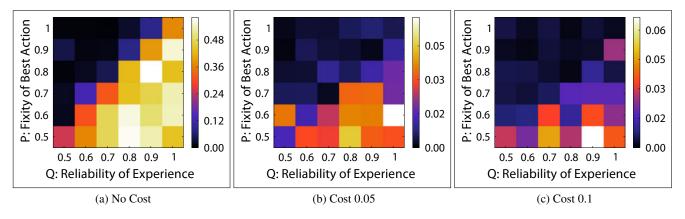


Figure 6: The evolved plasticity levels under the basic experimental setup. Colors correspond to evolved learning rates (η) . The colors are scaled differently for different figures, to emphasize their *internal* variations. – Results averaged over 20 runs

the case where learning is associated with a cost, but our results (Figure 6b and 6c) indicate that in this case, the border will shift, making the non-learning region larger.

The results for the individuals evolved with a cost of learning are quite noisy, due in part to the very low levels of learning effort that were evolved. The reason the levels can be so low is that, in this experiment, 49 presentations of stimuli are available to learn from *before* being tested. In the rest of our experiments, this simplifying assumption (of a long training period before testing) is removed.

Multiple decisions

Extending the experience to multiple decisions has these beneficial effects:

 It makes the experiment more realistic – natural organisms are likely to have their survival- and reproduction ability depend on several decisions rather than a single one. 2. It makes the evolutionary search proceed more efficiently, as it is now possible to identify intermediate fitness levels. In other words, there are more "stepping stones" for evolution to visit on the way to a good solution.

The results (Figure 7) are similar to those seen before, but more clean and systematic. In particular, the results for situations where plasticity is associated with a cost are different. In the basic setup, these plasticity levels were very low, but here, we see they have a magnitude comparable to the situation without costs. We now see clearly that the learning/no-learning border is shifted as a result of the cost.

Periodic changes

Changing the environment to contain *periodic changes* allows us to smoothly alter the best-action fixity within all values imaginable: from a completely stable environment to one that changes constantly. The periodic changes also imply that the environment has a *state* which the agent can

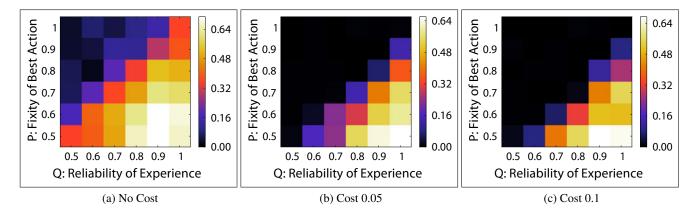


Figure 7: The evolved plasticity levels under the setup with multiple decisions. Colors correspond to evolved learning rates (η). The colors are scaled differently for different figures, to emphasize their *internal* variations. – Results averaged over 20 runs

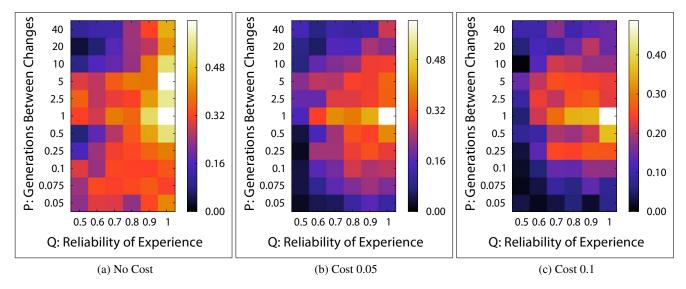


Figure 8: The evolved plasticity levels under the setup with periodic environmental changes. Colors correspond to evolved learning rates (η) . The colors are scaled differently for different figures, to emphasize their *internal* variations. The value p discussed here is defined differently than the one in other plots: It now signals the *average number of generations* between environmental changes. – Results averaged over 20 runs

learn. There was a state also in the previous environment, but it lasted only for a single experience-consequence phase pair. After one set of phases the environment was changed randomly. In other words, there was no reason for the agent to learn lasting associations – rather, it was encouraged to do learning only by the somewhat artificial variation we imposed between experience and consequence phases.

Evolved learning rates in this environment (Figure 8) are not directly comparable to the ones in the two previous experiments, because the separation between the experience-and consequence phase in those experiments made it impossible to realize an environment that was changing *too fast* to learn. The association observed in the experience phase would always last until the consequence phase, and could therefore always be learned.

Although the experiments are not directly comparable, we can observe some relationships between the basic experiment and the one with periodic changes. In the basic experiment, a p-value of 0.5 means the environment will have a 50 % chance of changing each generation. This corresponds roughly to a stability period of 1-2 generations. Increasing the stability period from 1 corresponds to increasing p in the basic experiment, making the best action more stable. As seen in Figures 6 and 7, this has the effect of lowering the evolved learning rates. The same is observed in Figure 8 for p-values increasing above 1. In fact, the top half of Figure 8a (the part above p = 1) is quite similar to Figures 6a and 7a.

In light of this comparison, we can see what the original experiment by Dunlap and Stephens was missing: environ-

ments with a lower stability period than a single generation. For such environments (Figure 8), decreasing the best-action fixity substantially will *select against* learning, as a result of learned associations more rapidly losing their utility. In other words, the benefits of learning decrease as the amount of time learned associations can be exploited decreases.

Summarizing, we can see that the *revised pq-model* (Figure 9) of learning has the following properties:

- 1. A range of best-action fixity levels select for learning. For environments where the best action changes too frequently or too infrequently, learning is selected against.
- 2. An increased reliability of experience selects for learning.
- 3. Plasticity levels increase the further away from the non-learning zone we are.

Conclusion

Through simulated evolution of learning individuals under different levels of environmental change we have studied and unified two seemingly conflicting models of the evolution of learning ability.

The *pq-model* captures two important properties of the evolution of learning: Learning requires 1) *environmental changes* and 2) *reliable experiences* to evolve. The *Goldilocks principle* model suggests that a *range of environmental variability* selects for learning, while both too stable and too variable environments select against it.

Initially, we saw results lending support to the pq-model for the original experiment it was designed for. We later

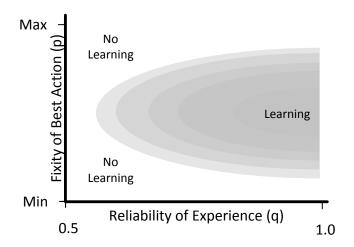


Figure 9: The revised pq-model (see Figure 1 for original) of how learning relates to environmental change. Darker shades of gray indicate higher levels of plasticity.

showed that the model also holds for more realistic scenarios and for scenarios where learning has a cost. Applying an explicit cost to learning ability moved the "neutral region" of learning, making more areas in the model select for non-learning strategies, as one might expect. To study the relationship between the pq-model and Goldilocks principle, we needed to generalize the experiment Dunlap and Stephens originally proposed and study the full range of environmental change, from full stability to constant changes. The results led us to suggest a *revised pq-model* (Figure 9).

In conclusion, we want to emphasize that this model by no means tells the full truth about the relationship between environmental variability and evolved learning efforts. The model shows one part of the truth, but it leaves out several issues known to be important to this relationship, such as age-varying learning efforts (Knudsen (2004)) and the complex interaction between agents and their environments (Beer (1995)). Further studies of these and other related topics will increase our understanding of how learning evolves, and how the evolved learning is influenced by environmental properties.

References

- Beer, R. D. (1995). A dynamical systems perspective on agentenvironment interaction. Artificial Intelligence, 72(1-2):173– 215.
- DeWitt, T. J., Sih, A., and Wilson, D. S. (1998). Costs and limits of phenotypic plasticity. *Trends in Ecology & Evolution*, 13(2):77–81.
- Dukas, R. (1998). Evolutionary ecology of learning. In *Cognitive ecology: The evolutionary ecology of information processing and decision making*, pages 129–174. University of Chicago Press, Chicago.

- Dunlap, A. S. and Stephens, D. W. (2009). Components of change in the evolution of learning and unlearned preference. *Proceedings of the Royal Society B: Biological Sciences*, 276(1670):3201–3208.
- Ellefsen, K. O. (2013). Balancing the Costs and Benefits of Learning Ability. In *Advances in Artificial Life, ECAL 2013: Proceedings of the Twelfth European Conference on the Synthesis and Simulation of Living Systems*, pages 292–299.
- Floreano, D. and Urzelai, J. (2001). Evolution of Plastic Control Networks. *Autonomous Robots*, 11(3):311–317.
- Hebb, D. O. (1949). *The Organization of Behavior*. Wiley and Sons, New York.
- Kerr, B. and Feldman, M. (2003). Carving the Cognitive Niche: Optimal Learning Strategies in Homogeneous and Heterogeneous Environments. *Journal of Theoretical Biology*, 220(2):169–188.
- Knudsen, E. I. (2004). Sensitive Periods in the Development of the Brain and Behavior. *Journal of Cognitive Neuroscience*, 16(8):1412–1425.
- Littman, M. (1995). Simulations Combining Evolution and Learning. In *Adaptive Individuals in Evolving Populations: Models and Algorithms: Santa Fe Institute Studies in the Sciences of Complexity*, pages 465–477. Addison-Wesley.
- Mayley, G. (1996). Landscapes, Learning Costs, and Genetic Assimilation. *Evolutionary Computation*, 4(3):213–234.
- Mery, F. and Kawecki, T. (2002). Experimental Evolution of Learning Ability in Fruit Flies. *Proceedings of the National Academy of Sciences*, 99(22):14274–14279.
- Nolfi, S. and Parisi, D. (1996). Learning to Adapt to Changing Environments in Evolving Neural Networks. *Adaptive Behavior*, 5(1):75–98.
- Nolfi, S., Parisi, D., and Elman, J. L. (1994). Learning and Evolution in Neural Networks. Adaptive Behavior, 3(1):5–28.
- Sasaki, T. and Tokoro, M. (1999). Evolving Learnable Neural Networks under Changing Environments with Various Rates of Inheritance of Acquired Characters: Comparison between Darwinian and Lamarckian Evolution. *Artificial Life*, 5(3):203–223.
- Soltoggio, A., Dürr, P., Mattiussi, C., and Floreano, D. (2007). Evolving neuromodulatory topologies for reinforcement learning-like problems. In *Proceedings of the IEEE Congress on Evolutionary Computation* 2007, pages 2471–2478. IEEE.
- Todd, P. M. and Miller, G. F. (1991). Exploring adaptive agency II: Simulating the evolution of associative learning. In From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior, pages 306– 315, Cambridge, MA. MIT Press/Bradford Books.
- Watson, J. and Wiles, J. (2002). The rise and fall of learning: a neural network model of the genetic assimilation of acquired traits. In *Proceedings of the 2002 Congress on Evolutionary Computation.*, pages 600–605. IEEE.

Evolution of Hierarchical Controllers for Multirobot Systems

Miguel Duarte, Sancho Moura Oliveira and Anders Lyhne Christensen

Instituto de Telecomunicações &
Instituto Universitário de Lisboa (ISCTE-IUL), Lisbon, Portugal
{miguel_duarte, sancho.oliveira, anders.christensen}@iscte.pt

Abstract

Decentralized control for multirobot systems is difficult to design by hand because the behavioral rules for individual robots cannot, in general, be derived from a desired collective behavior. System designers have therefore resorted to evolutionary computation as a means to heuristically synthesize self-organized behaviors for robot collectives. Evolutionary computation is typically applied by putting the rules governing the individual robots under evolutionary control and by assigning fitness scores based on collective performance. Scaling evolutionary approaches to complex tasks has, however, proven challenging due to issues related to bootstrapping and premature convergence. In this paper, we show how hierarchical task decomposition and the combination of evolved and preprogrammed control can overcome these issues. We apply our approach to a complex multirobot task that requires a high degree of coordination and collective decision making, and we synthesize controllers capable of solving the task.

Introduction

One of the major challenges in a multirobot system (MRS) is coordination. In a MRS, the behavior of a robot depends not only on interactions with the immediate environment but also on the behavior of other robots. While centralized control can facilitate coordination, it cannot be used in all scenarios due to communication and/or computational constraints on the robots (Crespi et al., 2008). On the other hand, decentralized control is often difficult to design by hand because the behavioral rules for individual robots cannot be derived from a desired macroscopic behavior (Dorigo et al., 2004). Several researchers have taken inspiration from nature when designing large-scale distributed MRS, such as the use of pheromones (Payton et al., 2003) and division of labor in social insects (Labella et al., 2006), coordination in bacteria, and aggregation of amoeba into slime mold (Şahin, 2005). For many multirobot tasks, we may, however, be unable to find examples in nature from which to draw inspiration, and other means to design decentralized control are therefore necessary.

In the field of evolutionary robotics (ER), evolutionary computation is applied to synthesize control and sometimes the morphology of autonomous robots (Nolfi and Floreano, 2000). Such approaches have the potential to automate the design of self-organized behavior without manual specification of the microscopic rules that govern each individual (Floreano and Keller, 2010). The experimenter only has to define the high-level collective objective as a fitness function, and an evolutionary process optimizes the individual rules with respect to that function.

In the domain of MRS, controllers have been evolved to, for instance, enable a swarm of robots to establish and maintain aerial data links in natural disaster scenarios (Hauert et al., 2009), to play robotic soccer (Uchibe and Asada, 2006), and to collectively transport heavy objects (Groß and Dorigo, 2009). The *online* (or *embodied*) (Floreano and Mondada, 1998; Watson et al., 1999; Silva et al., 2012) evolution of controllers, where evolution takes place during task execution, has furthermore been demonstrated for simple tasks such as aggregation.

Although evolutionary techniques have been shown capable of synthesizing surprisingly simple and elegant controllers for MRS tasks (Mosteo and Montano, 2010), it has proven difficult to bootstrap the evolutionary process when solutions to complex tasks are sought (Nelson et al., 2009). Different incremental approaches have been studied as a means to overcome the bootstrapping problem. Whiteson et al. (2005) experimented with task decomposition and decision trees for keepaway soccer. The authors evolved artificial neural networks (ANN) for basic tasks, such as passing the ball to a teammate and intercepting the ball. Decision nodes were then manually programmed to choose which ANN should be active. Christensen and Dorigo (2006) compared two different incremental evolutionary approaches in a multirobot phototaxis and hole-avoidance task. They found no benefits over a non-incremental approach when compared with an incremental approach where the controllers were trained on different sub-tasks sequentially, or an incremental increase in environmental complexity.

Novelty search (Lehman and Stanley, 2011) has recently been applied to the evolution of behaviors for swarms of robots (Gomes et al., 2013) as a means to overcome bootstrapping issues and to avoid premature convergence. In novelty search, behaviors are scored based on their novelty with respect to previously evaluated behaviors, and not based on a traditional fitness function. Given the lack of a static objective in novelty search, bootstrapping is typically not an issue, and the constant evolutionary pressure toward behavioral innovation means that novelty search is unaffected by deception. However, as shown by Gomes et al. (2013), the choice of behavior characterization has a significant impact on the performance of novelty search. For non-trivial tasks, it may be difficult to design effective behavioral characterizations that lead to the evolution of good solutions.

In this paper, we show how evolutionary computation can be combined with preprogrammed control in an engineering-centric approach (Duarte et al., 2014a) for the semi-automatic synthesis of controllers for complex multirobot tasks. We recursively decompose the goal task into sub-tasks and synthesize different controllers (either evolved or preprogrammed) to solve each sub-task. The controllers for the sub-tasks are then combined hierarchically in order to solve the goal task. Our hierarchical decomposition approach has been shown to successfully synthesize controllers for complex, single-robot tasks, as well as enabling controllers to be sucessfully transfered to real hardware (Duarte et al., 2014a). In this study, we demonstrate how our approach can be scaled to multirobot tasks, and we compare our hierarchical approach to two non-hierarchical approaches: a simple generational evolutionary algorithm and NEAT (Stanley and Miikkulainen, 2002).

Methodology

Our objective is to exploit the capacity of artificial evolution to automatically synthesize behavior in the design of controllers for complex, real robot tasks. We manually divide a task into simpler sub-tasks when the evolutionary process is unable to find a solution. Sub-controllers are evolved to solve each sub-task, and the resulting controllers are then composed in a bottom-up, hierarchical fashion through the evolution of one or more higher-level controllers.

The complete controller for a complex task is composed of several sub-controllers arranged in a hierarchy (see Figure 1). Nodes in the hierarchy are either *behavior arbitrators* or *behavior primitives* (a similar terminology is used in (Lee, 1999)), and can be synthesized with different techniques, such as evolution of ANNs, genetic programming, fuzzy logic or manual programming. Behavior primitives are at the bottom of the controller hierarchy and directly control the actuators of the robot, such as wheels. Each robot in the MRS has its own independent copy of the controller. Different robots may therefore be executing different behavior primitives at any given time.

A behavior primitive is evolved to solve a task if an *appropriate fitness function* can be found. An appropriate fitness

function (i) allows evolution to bootstrap, (ii) leads evolution to controllers that are able to successfully solve the task, and (iii) leads evolution to controllers that are able to maintain performance when transferred to real robots. If it is not possible to find such a fitness function, the task is manually and recursively divided into sub-tasks until an appropriate fitness function has been found for each sub-task.

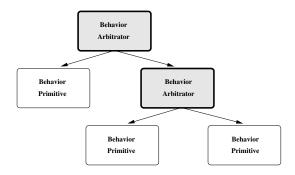


Figure 1: A representation of the hierarchical controller running on each robot in a MRS. A behavior arbitrator delegates the control of the robot to one of its sub-controllers. A behavior primitive controls the actuators of the robot directly.

After the behavior primitives have been synthesized, they are combined through the evolution of a behavior arbitrator. A behavior arbitrator is responsible for delegating control to one or more of its sub-controllers. A sub-controller can be composed of several hierarchical levels of behavior arbitrators and primitives. Each behavior arbitrator receives either all or a subset of the robot's sensory inputs. A behavior arbitrator might have access to some specific sensors that other behavior arbitrators and behavior primitives do not need. In the study presented in this paper, the behavior arbitrators are ANN-based and select only one of their sub-controllers at any given time, namely the one corresponding to the output neuron with the highest activation, but other arbitration methods could be used.

Some tasks may require the robots to perform actions that are too difficult to simulate accurately, or may be challenging to decompose into sub-tasks. In these particular cases, basic preprogrammed behavior primitives can be included in the controller hierarchy as we demonstrate in the Section titled "Evolving a Hierarchical Controller".

Experimental Setup

We apply hierarchical task decomposition to a generic object removal task, which was designed to require several behaviors typically associated with ER in MRS, namely collective search and navigation (Quinn et al., 2003), obstacle avoidance (Christensen and Dorigo, 2006), and collective decision-making (Floreano et al., 2007). We explore a more complex version of the task presented in (Ollion and Doncieux, 2011), in which a group of robots must individu-

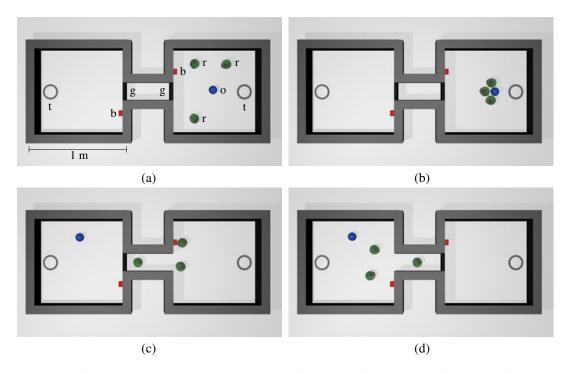


Figure 2: An example of the robots executing the task, where \mathbf{r} is a robot, \mathbf{o} is an object, \mathbf{g} is a gate, \mathbf{b} is a button, and \mathbf{t} is a basket. (a): the robots start the task randomly distributed in one of the rooms. (b): the robots aggregate around the object and push it toward the basket. (c): the robots enter the corridor after one of them has pushed the button. (d): the robots enter the other room in order to find the new object.

ally transport balls into a basket. In our version of the task, the environment is composed of two rooms, connected by a corridor. The corridor is blocked by two gates, one at each entrance, which can be opened by pushing a button (see Figure 2). The rooms are rectangular and side lengths vary between 75 cm and 150 cm drawn from a uniform distribution. The robots must find and remove an object by pushing it into a basket (see Figure 2a-b). The object can only be pushed if all robots cooperate. When the robots succeed at pushing the object into the basket, the object is removed and a new object is automatically placed in the other room. To traverse the corridor connecting the two rooms, one of the robots must first push a button to open the gates. Once the button has been pushed, the gate closer to the robots will remain open until all robots have entered the corridor (see Figure 2c). Once all the robots have entered the corridor, the first gate will close and the gate leading to the other room will open (see Figure 2d).

We use three simulated differential-drive robots, loosely modeled after the e-puck robot (Mondada et al., 2009). The robots are circular with a diameter of 7.5 cm, and they are equipped with the following sensors: (i) four infrared (IR) sensors for obstacle detection, with a range of 12 cm, (ii) four sensors to detect the object that should be removed, with a range of 30 cm, (iii) four sensors to detect the basket, with a range of 100 cm, (iv) four sensors to de-

tect nearby robots, with a range of 100 cm, (v) one sound sensor, and (vi) two sensors to detect the gate button, positioned at -30° and 30° with respect to the front of the robot. The gate button sensors have a range of 100 cm and an opening angle of 120° . The wall, object and robot sensors are positioned at intervals of 90° around the perimeter of the robot (front, back, left, and right). The set of actuators is composed of two wheels, which allow the robots to move at speeds of up to 10 cm/s. The simulated sensors have been modelled based on data collected from a real e-puck's sensors, and have been previously used to successfully transfer controllers from simulation to a real e-puck robot (Duarte et al., 2014a).

For evolved behavior arbitrators and behavior primitives, we use continuous-time recurrent neural networks (Beer and Gallagher, 1992). In simulation, we run a total of ten evolutionary runs with a population size of 100 genomes for each node in the controller hierarchy. The fitness score assigned to each genome is the mean score obtained in 30 simulations with different initial conditions. The five highest scoring genomes are copied directly to the next generation. Another 19 copies of each genome are made and mutation is applied to each gene with a probability of 10%. A Gaussian offset with a mean of 0 and a standard deviation of 1 is applied when a gene undergoes mutation. After the evolutionary process has terminated, we run a post-evaluation

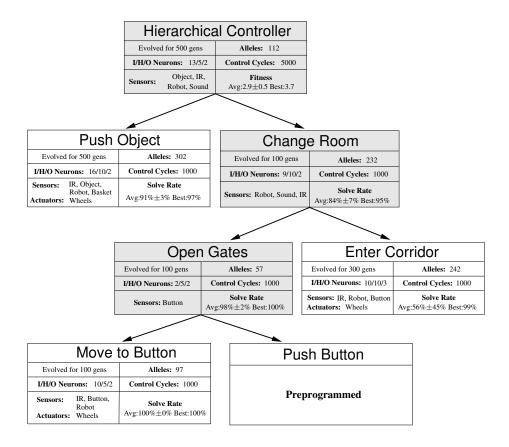


Figure 3: The complete controller for the two-room object removal task is composed of three behavior arbitrators and four behavior primitives.

with 100 samples to obtain a more accurate estimate of the controllers' performance. To obtain more general and adaptive behaviors, the number of robots used during the evolutionary process of the behavior primitives and arbitrators was varied from one to three.

The fitness functions used in this study are functional incremental fitness functions (Nelson et al., 2009). Each fitness function typically has a number of cases that represent different outcomes of a simulation sample. Each case is kept simple and typically has only one or two terms. The cases are weighted by multiplying by a factor and by adding a constant, initially set to values of one and zero, respectively. The values of the constants and factors are adjusted through an empirical trial-and-error process when necessary: if a bootstrapping case is exploited over a goal case, guiding evolution toward local optima, the relative weight of the exploited case is either decreased or the weight of the other cases are increased. Given the simplicity of the fitness functions used, this process usually only required a couple of iterations.

We use JBotEvolver (Duarte et al., 2014c), an open source, multirobot simulation platform and neuroevolution framework, for the experiments presented in this paper. JBotEvolver allows to automate the main steps of the hi-

erarchical composition process. The experimenter defines a configuration file with the various sub-controllers, including information on how they are connected and the evolutionary setup for each node. The simulator then synthesizes and composes the hierarchical controller in a bottom-up fashion by conducting several evolutionary runs (typically ten) for one node in the hierarchy at a time, and by selecting the ANN displaying the highest performance for the node. The simulator, configuration files and results can be found online (Duarte et al., 2014b).

Evolving a Hierarchical Controller

We applied our hierarchical decomposition methodology in order to synthesize controllers for the object removal task. Figure 3 provides an overview of the final controller, as well as the parameters of the neural network used, sensors, actuators, and performance achieved by each sub-controller. We started by decomposing the task into two different subtasks: "Push Object" and "Change Room". An appropriate fitness function for the "Push Object" sub-task could easily be found (see the Section "Finding and pushing the object"), and we evolved a behavior primitive for that sub-task. The "Change Room" sub-task, on the other hand, proved more

challenging, and we therefore decomposed the sub-task into two new sub-tasks: "Open Gates" and "Enter Corridor". The "Open Gates" sub-task was further decomposed into a "Move to Button" evolved behavior primitive and a "Push Button" preprogrammed behavior primitive. In the following subsections, we discuss each sub-task in detail.

Finding and pushing the object

We started with the "Push Object" behavior primitive, which was evolved in a single room without any gates or corridor. The robots had to collectively find and push a randomly placed object into a basket (see Figure 2b). As soon the object was within the boundaries of the basket, the sub-task was considered solved and the simulation was terminated. The fitness function f_1 used for this sub-task rewards robots for quickly pushing the object toward the basket:

$$f_1 = \begin{cases} 2 - \frac{c}{C} & \text{if object pushed to the basket} \\ \\ \frac{D - d}{D} & \text{otherwise} \end{cases}$$

where c is the number of cycles spent by the robots to successfully push object to the basket, C is the maximum length of a simulation run (1000 control cycles), D is the initial distance from the object to the basket, and d is the final distance from the object to the basket. The fitness trajectories for the "Push Object" behavior primitive can be seen in Figure 4.

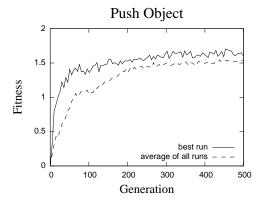


Figure 4: Fitness trajectories for "Push Object' primitive, with the best fitness and the mean fitness for each generation.

Opening the gates

For the group of robots to cross from one room to the other, one of the robots has to push a button, which opens the gates to the corridor (see Figure 2c-d). If a robot correctly pushes the button, the gate opens and emits a noise for two seconds that can be heard by all robots. The buttons are only 2.5 cm in diameter, and they must be pushed by hitting them at an

angle steeper than approximately 45°. Pushing a button to open the gates requires fine sensorimotor coordination, since the buttons are difficult to detect and hit. This is a difficult interaction to model correctly in simulation, and it can also be a challenging behavior to evolve and transfer successfully to real robotic hardware. We therefore decided to manually program a controller to push the button, using the robot's camera to find and move to the button. In simulation, however, activating the preprogrammed behavior automatically opens the gate. When the preprogrammed behavior primitive is activated, the control cycle of the main behavior arbitrator is stopped. While we did not attempt to transfer the evolved controllers in the experiments presented in this paper, the preprogrammed "Push Button" behavior was essential in order to cross the reality gap in previous experiments with a single e-puck robot (Duarte et al., 2014a).

We evolved the "Move to Button" behavior primitive, where the controller was evaluated according to a distance gradient to the button, with a time-dependent bonus upon correctly completing the task. The fitness function used for this sub-task is defined below:

$$f_2 = \begin{cases} 1 + \frac{C - c}{C} & \text{if achieved objective} \\ \\ \frac{D - d}{D} & \text{otherwise} \end{cases}$$

where C the maximum length of a simulation run (1000 control cycles), c is the number of cycles spent, D is the distance from the robots' average starting point to the button, and d is the shortest distance to the objective that any robot reached. In this case, the objective is to get within 5 cm of the button.

Afterwards, we evolved the "Open Gates" behavior arbitrator, which has access to the evolved "Move to Button" behavior primitive and to the preprogrammed "Push Button" behavior primitive (see Figure 3). The controller was evaluated using f_2 , where the objective was to open the gate leading to the corridor (see Figure 2b-c). The fitness trajectories for the "Move to Button" behavior primitive and the "Open Gates" behavior arbitrator can be seen in Figure 5. As it can be seen in the figure, good solutions to the "Open Gates" are found in the first generation. This result is explained by the low difficulty of the sub-task: the controller only has to change from the "Move to Button" behavior primitive to the "Push Button" behavior arbitrator when the robot is close to the button.

Crossing from one room to the other

After pushing the button, the robots should enter the corridor in order to move into the other room (see Figure 2c). We evolved the "Enter Corridor" behavior primitive, which has to move the robot inside the corridor after the first gate has been opened. The controller should then move the robot inside the other room when the second gate opens (see Fig-

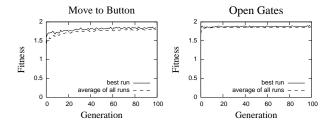


Figure 5: Fitness trajectories for "Move to Button" primitive and the "Open Gates" arbitrator.

ure 2d). Each controller was evaluated according to the following fitness function:

$$f_3 = \begin{cases} 2-10^{-3} \cdot w & \text{if robots reached the other room} \\ \frac{D-d}{D} & \text{otherwise} \end{cases}$$

where w is the number of times the robots collided with a wall, D is the distance from the robots' average starting point to the end of the corridor, and d is the point closest to the end of the corridor that the robots reached.

We then evolved the "Change Room" sub-controller, which is composed of three behavior primitives and two behavior arbitrators (see Figure 3). For the evolution of the "Change Room" behavior arbitrator, the robots were positioned and oriented randomly in one of the rooms at the beginning of each trial, one robot had to push the button, and all robots then had to enter the corridor and move to the other room. The "Change Room" controller was evaluated using f_3 . The fitness trajectories for the "Enter Corridor" behavior primitive and the "Change Room" behavior arbitrator can be seen in Figure 6.

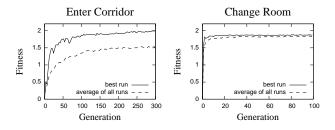


Figure 6: Fitness trajectories for "Enter Corridor" behavior primitive and the "Change Room" behavior arbitrator.

The Hierarchical Controller arbitrator

In the complete task, the first object can appear in any of the two rooms and the robots should remove as many objects as possible. To obtain a controller for the complete task, the "Push Object" behavior primitive and the "Cross Rooms" behavior arbitrator were combined via the evolution of a "Hierarchical Controller" behavior arbitrator (see Figure 3). All evolutionary runs converged to a behavior where the robots would move between rooms whenever they removed an object. The controllers were evaluated according to the number of objects they removed, with a small bonus for getting collectively closer to the object and a small penalty for colliding with walls and other robots, according to f_4 :

$$f_4 = P + \frac{D-d}{D} - 10^{-3} \cdot w$$

where P is the number of objects successfully pushed into the basket, D is the initial distance of the last object to the basket, d is the final distance of the last object to the basket, and w is the number of times a robot collided with a wall.

The controllers were evolved with a time limit of 5000 control cycles (equivalent to 500 seconds), and the highest performing controller was able to remove a mean of 3.74 objects, which corresponds to one object every 2 minutes and 14 seconds on average. The average number of removed objects from the highest performing controllers in each of the 10 experimental runs was 2.98 ± 0.52 , which corresponds to an average of 2 minutes and 48 seconds per object. The fitness trajectory "Hierarchical Controller" behavior arbitrator can be seen in Figure 7.

Comparison with non-hierarchical approaches

We compared the results obtained with our hierarchical approach with two non-hierarchical approaches: a classic evolutionary robotics setup, with a simple generational algorithm, and a setup where controllers were evolved using NEAT (Stanley and Miikkulainen, 2002). The NEAT implementation available in the open-source Encog Machine Learning Framework (Heaton, 2008) was integrated in our simulator platform JBotEvolver, and we used the parameter values proposed in (Stanley and Miikkulainen, 2002). Since the controllers in these setups are unable to use the "Push Button" preprogrammed behavior primitive, we introduced a "Button Actuator" that allows the robot to open the gates.

The controllers evolved in the respective setups were only able to partly solve the task. In the classic setup, controllers from the six out of ten evolutionary runs learned how to push the object into the basket when the object was initially positioned in the same room as the robots. The mean fitness achieved in the ten evolutionary runs was 0.3 ± 0.2 , and the highest fitness achieved was 0.5. No controller was able to successfully open the gates and navigate to the other room.

Controllers in all ten evolutionary runs in the NEAT setup learned how to push the object into the basket. However, the controllers evolved with NEAT were also unable to cross to the other room, and moved in circles looking for objects even if there was none in the current room. The mean fitness

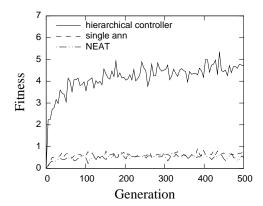


Figure 7: Fitness trajectories of the best fitness for each generation for the "Hierarchical Controller" arbitrator, the single-network setup, and the NEAT setup.

of all ten evolutionary runs was 0.4 ± 0.1 , and the fitness of the best evolutionary run was 0.5. The fitness trajectories of both single-network setups and the "Hierarchical Controller" behavior arbitrator setup are shown in Figure 7.

Even though the complete hierarchical controller has a total of six evolved behavior arbitrators and behavior primitives, each node in the hierarchy needed only to evolve for relatively few generations using a simple generational evolutionary algorithm. Fitness scores in some of the simpler nodes plateaued well before the end of the evolutionary run, and could have even been evolved for fewer generations (see Figures 4, 5 and 6). The total number of generations in the complete controller was 1600, and the mean number of generations per evolved node was 267 (see Figure 3).

Conclusions

In this paper, we showed how hierarchical task decomposition could enable the use of evolutionary synthesis of behavioral control for a complex MRS task. The robots had to collectively locate and push objects into a basket in an environment with two rooms connected by a corridor. In order to cross from one room to the other, one of the robots had to push a button that opened a gate leading to the corridor. We divided the task into multiple sub-tasks, and whenever a solution to a sub-task could not be evolved due to bootstrapping or premature convergence issues, the sub-task was further divided until solutions could be evolved for each sub-task. Finally, we combined the sub-task solutions through the evolution of behavior arbitrators.

Hierarchical decomposition of tasks and control has two key benefits: (i) it allows for the use of evolutionary computation to synthesize control based on self-organization in tasks that would otherwise be too complex, and (ii) individual controller components can be evolved under different conditions, or even be preprogrammed when a sub-task is too difficult to simulate accurately.

The application of our hierarchical approach depends on the capability of dividing a task into different sub-tasks or behaviors. This division, however, is only enforced if traditional evolutionary techniques are unable to find adequate solutions for a given task. While state-of-the-art techniques such as novelty search (Lehman and Stanley, 2011) may be able to evolve solutions for more complex tasks than simpler evolutionary algorithms such as the one used in this study, it is reasonable to assume that even such approaches have limits in terms of task complexity. Modern neuroevolution algorithms could, however, be used in combination with hierarchical decomposition to allow for the synthesis of controllers with fewer, more complex behavior primitives.

We are currently studying the use of state-of-the-art neuroevolution algorithms in hierarchical controller synthesis, as well as the transfer of hierarchically evolved controllers for complex multirobot tasks to real robotic hardware.

Acknowledgements

This work was partly supported by FCT – Foundation of Science and Technology under grants SFRH/BD/76438/2011, PEst-OE/EEI/LA0008/2013 and EXPL/EEI-AUT/0329/2013.

References

- Beer, R. D. and Gallagher, J. C. (1992). Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, 1(1):91–122.
- Christensen, A. L. and Dorigo, M. (2006). Evolving an integrated phototaxis and hole avoidance behavior for a swarm-bot. In *Proceedings of 10th International Conference on the Simulation & Synthesis of Living Systems (ALIFE)*, pages 248–254. MIT Press, Cambridge, MA.
- Crespi, V., Galstyan, A., and Lerman, K. (2008). Top-down vs bottom-up methodologies in multi-agent system design. *Autonomous Robots*, 24(3):303–313.
- Şahin, E. (2005). Swarm robotics: From sources of inspiration to domains of application. In *Proceedings of the Swarm Robotics Workshop*, pages 10–20. Springer, Berlin, Germany.
- Dorigo, M., Trianni, V., Şahin, E., Groß, R., Labella, T., Baldassarre, G., Nolfi, S., Deneubourg, J., Mondada, F., Floreano, D., and Gambardella, L. M. (2004). Evolving self-organizing behaviors for a swarm-bot. *Autonomous Robots*, 17(2):223–245.
- Duarte, M., Oliveira, S. M., and Christensen, A. L. (2014a). Evolution of hybrid robotic controllers for complex tasks. *Journal of Intelligent and Robotic Systems*. In press.

- Duarte, M., Oliveira, S. M., and Christensen, A. L. (2014b). Simulator, configuration files and results. URL http://home.iscte-iul.pt/~alcen/hierarchicalmultirobot.
- Duarte, M., Silva, F., Rodrigues, T., Oliveira, S. M., and Christensen, A. L. (2014c). JBotEvolver: A versatile simulation platform for evolutionary robotics. In *Proceedings of the 14th International Conference on the Synthesis & Simulation of Living Systems (ALIFE)*. MIT Press, Cambridge, MA. In press.
- Floreano, D. and Keller, L. (2010). Evolution of adaptive behaviour in robots by means of Darwinian selection. *PLoS Biology*, 8(1):1–8.
- Floreano, D., Mitri, S., Magnenat, S., and Keller, L. (2007). Evolutionary conditions for the emergence of communication in robots. *Current biology*, 17(6):514–519.
- Floreano, D. and Mondada, F. (1998). Evolutionary neuro-controllers for autonomous mobile robots. *Neural Networks*, 11(7–8):1461–1478.
- Gomes, J., Urbano, P., and Christensen, A. L. (2013). Evolution of swarm robotics systems with novelty search. *Swarm Intelligence*, 7(2-3):115–144.
- Groß, R. and Dorigo, M. (2009). Towards group transport by swarms of robots. *International Journal of Bio-Inspired Computing*, 1(1–2):1–13.
- Hauert, S., Zufferey, J.-C., and Floreano, D. (2009). Evolved swarming without positioning information: an application in aerial communication relay. *Autonomous Robots*, 26(1):21–32.
- Heaton, J. (2008). Encog machine learning framework. URL http://www.heatonresearch.com/encog.
- Labella, T. H., Dorigo, M., and Deneubourg, J.-L. (2006). Division of labor in a group of robots inspired by ants' foraging behavior. ACM Transactions on Autonomous and Adaptive Systems, 1(1):4–25.
- Lee, W.-P. (1999). Evolving complex robot behaviors. *Information Sciences*, 121(1-2):1–25.
- Lehman, J. and Stanley, K. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223.
- Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C., Floreano, D., and Martinoli, A. (2009). The e-puck, a robot designed for education in engineering. In *Proceedings of* the 9th Conference on Autonomous Robot Systems and Competitions (ROBOTICA), pages 59–65. Instituto Politecnico de Castelo Branco, Castelo Branco, Portugal.

- Mosteo, A. and Montano, L. (2010). A survey of multirobot task allocation. Technical report, Instituto de Investigación en Ingeniería de Aragón, Universidad de Zaragoza, Saragossa, Spain.
- Nelson, A. L., Barlow, G. J., and Doitsidis, L. (2009). Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57(4):345–370.
- Nolfi, S. and Floreano, D. (2000). *Evolutionary robotics:* The biology, intelligence, and technology of self-organizing machines. MIT press, Cambridge, MA.
- Ollion, C. and Doncieux, S. (2011). Why and how to measure exploration in behavioral space. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation Conference (GECCO)*, pages 267–274. ACM, New York, NY.
- Payton, D., Estkowski, R., and Howard, M. (2003). Compound behaviors in pheromone robotics. *Robotics and Autonomous Systems*, 44(3):229–240.
- Quinn, M., Smith, L., Mayley, G., and Husbands, P. (2003). Evolving controllers for a homogeneous system of physical robots: Structured cooperation with minimal sensors. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 361(1811):2321–2343.
- Silva, F., Urbano, P., Oliveira, S., and Christensen, A. L. (2012). odNEAT: An algorithm for distributed online, onboard evolution of robot behaviours. In *Proceedings of the 13th International Conference on the Simulation & Synthesis of Living Systems (ALIFE)*, pages 251–258. MIT Press, Cambridge, MA.
- Stanley, K. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.
- Uchibe, E. and Asada, M. (2006). Incremental coevolution with competitive and cooperative tasks in a multirobot environment. *Proceedings of the IEEE*, 94(7):1412–1424.
- Watson, R., Ficici, S., and Pollack, J. (1999). Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC)*, pages 335–342. IEEE Press, Piscataway, NJ.
- Whiteson, S., Kohl, N., Miikkulainen, R., and Stone, P. (2005). Evolving keepaway soccer players through task decomposition. *Machine Learning*, 59(1):5–30.

The Kinetic Basis of Self-Organized Pattern Formation

Yuri Shalygo¹

¹Gamma Ltd, Vyborg, Russia yuri.shalygo@gmail.com

Abstract

In his seminal paper on morphogenesis (1952), Alan Turing demonstrated that different spatio-temporal patterns can arise due to instability of the homogeneous state in reaction-diffusion systems, but at least two species are necessary to produce even the simplest stationary patterns. This paper is aimed to propose a novel model of the analog (continuous state) kinetic automaton and to show that stationary and dynamic patterns can arise in one-component networks of kinetic automata. Possible applicability of kinetic networks to modeling of real-world phenomena is also discussed.

Introduction

Pattern formation has been a well-investigated research field since the pioneering work of Alan Turing on morphogenesis (Turing, 1952). He showed that pattern formation can be accomplished by the interaction of two substances that spread with different rates. For the interactions of this type, Turing introduced the term reaction-diffusion systems, which is now generally in use. He demonstrated that a homogeneous steady state is unstable in certain such systems, and any small local deviation from this steady state is sufficient to trigger the onset of pattern formation. This phenomenon, termed diffusion-driven instability, has been found to be applicable in many biological and chemical systems and was analyzed mathematically using a variety of techniques (Murray, 1993).

The classical Turing instability leads to the establishment of stationary spatial patterns. However, the oscillatory analogue of this instability is possible and it was also envisaged by Alan Turing. This oscillatory instability produces traveling or standing waves and therefore it is often called the wave instability (Walgraef, 1997). Turing suggested that at least three species are needed for the oscillatory instability. The oscillatory instability is much rarer and has been found only in some chemical systems (Zhabotinsky et al., 1995) and biological phenomena (Meinhardt, 2004).

Owing to Turing's indisputable authority, it is often considered as a well-established fact that patterns can arise only in multicomponent dynamical systems. Actually, it holds mainly for reactive systems with passive transport, or, more precisely, chemical continuum media with gradient driven diffusion, described by Fick's Law and explained by Einstein in 1905 under the random walk assumption. However, an increasing number of natural phenomena, which are called sub-diffusion, super-diffusion or anomalous diffusion in general (Klafter and Sokolov, 2005), do not fit this relatively simple description of diffusion. From the signaling of

biological cells to the foraging behaviour of animals, the overall motion of objects is better described by steps that are not independent and can take vastly different time to perform. Some systems can be modeled as networks of decision-making agents reacting to external events or signals and can be regarded as reflexive systems with active transport, where the direction of motion depends not only on gradients but on many other factors, which may lead to "The rich get richer" phenomenon.

This paper is aimed to propose a novel numerical algorithm, called *Conservative Rank Transform* and an analog model of an abstract autonomous agent, called a *kinetic automaton* or a *kinon* for short; then to show that different dynamical patterns found in Turing multi-component systems can arise even in one-component *kinetic networks*, defined as *reflexive dynamical systems with active transport*.

Background

The proposed model stems from the cellular automata (CA) framework, conceived in the early 1950's by J. Von Neumann (Neumann, 1960) and Stanislaw Ulam (Ulam, 1952) and became popular among researchers largely due to John Conway's Game of Life (Gardner, 1970). The popularity of CA can be attributed to the enormous potential they hold in modeling complex systems and their simplicity, which is determined by the regularity of an underlying lattice and a fixed number of cell states. Frisch, Hasslacher and Pomeau (Frisch et al., 1986) and Wolfram (Wolfram, 1986) independently discovered that a simple cellular automaton on a 2D triangular lattice can simulate the Navier-Stokes equations and proposed an FHP model or Lattice Gas Automata (LGA), as these models are usually termed. Closely related to CA, Random Boolean Networks (RBN) were introduced by Stuart Kauffman as a model of genetic regulatory networks (Kauffman, 1969). It has been shown that Boolean idealization may capture the dynamics of genetic regulatory systems (Kauffman, 1993), but in general, Boolean approximation is inappropriate for modeling flow processes, e.g., movement of money through an economy, electricity over a grid, concentrations of metabolites in cell tissues, etc.

To address this issue, Coupled Map Lattices (CML) were proposed by Kunihiko Kaneko as a paradigm for the study of spatio-temporal complexity such as turbulence, convection, flows, population dynamics, etc. (Kaneko, 1985). CML can be viewed as a generalization of CA in terms of continuous state space and arbitrary network topology, but despite promising universality, CMLs have not become widespread.

On the contrary, Lattice Boltzmann Model (LBM), originally evolved from LGA and based on a minimal kinetic Boltzmann equation (Wolf-Gladrow, 2000), is attracting growing popularity. In LBM, representative particles ('parcels of fluid') evolve on a regular grid in accordance with simple streaming and collision rules designed to preserve fluid dynamics.

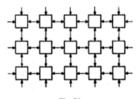
In the recent years, a lot of research has been directed to the continualization of the state space of different models. It was shown by Ulam that many deterministic problems in mathematics and physics can be converted into equivalent random processes and described by probabilities, which are real numbers (Ulam, 1952). The superiority of continuous values is confirmed by the mathematical theory of computable numbers and computable functions. It has been proved that simple analog (continuous state) computers can compute numbers and functions which are not computable by digital computers (Pour-El and Richards, 1989).

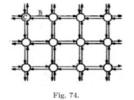
Despite the shown diversity, almost all existing modeling paradigms are derived from the cellular automaton model and inherit its restriction to lattice uniformity and a discrete state space. In some cases it leads to excessive oversimplification. According to Albert Einstein, "Everything should be made as simple as possible, but no simpler." The cellular automaton paradigm appeared in the very beginning of the computing era, when simplicity was compulsory. Modern computer technologies, in which even mobile phones outperform the former mainframes, pose new challenges to the modeling science. A new generation of topology and state space invariant modeling paradigms is needed. They will be inevitably more structured, but not necessarily much more complex. This work is a trial step in this direction.

Motivation

To a great extent this work was initiated by the now almost forgotten ideas of Konrad Zuse and Gordon Pask. Konrad Zuse, a German engineer who was the first to suggest that the entire universe is being computed on a computer, possibly a cellular automaton called "Rechnender Raum" or Calculating Space. In his paper he also gave an outline of a more advanced model called a net automaton (Zuse, 1969):

"Cellular automaton provides an elegant solution when each cell contains a complete calculating system, as symbolically represented in Figure 73. These single calculating systems contain both information-processing and information-storing elements. ... The net automaton represented in Figure 74 is a further development of the cellular automaton corresponding to Figure 73. The individual cells are responsible here for only information processing. Branching lines B connect the individual cells and serve both for information transmission and for information storage."





In his subsequent book on the theory of net automata (Zuse, 1975), which seems to have never been translated from German, Zuse considered mainly topological issues of the net automaton, but its internal structure was not elaborated. A decade earlier, Gordon Pask, a British cybernetician created a number of maverick machines, worked on the electrochemical device now known as Pack's Ear (Cariani, 1993). Pask demonstrated that such a device was able to construct its own sensors and effectors without having programmed them into a preset purpose. Using the same approach, he introduced an evolutionary model containing a diffusion network, which can be regarded as a precursor of the kinetic networks considered further (Pask, 1961):

"All a mean by a diffusion network is a system of tubes and basins, say, over which we can define food neighborhoods. A formal representation of a food diffusion network is shown in Fig. 1. It is a directed graph with nodes. The lines connecting nodes have quantities associated with them that represent the food impedance, the amount of resistance to the passage of food between nodes."

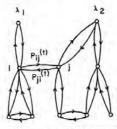


Fig. 1. Formal representation of food distribution network.

Despite the seeming disparity, both Zuse's net automaton and Pask's diffusion network have the central idea that nodes of the network are connected reciprocally with lines, having not only transport but also processing functions, and thus being active. This idea is in sharp contrast with the conventional view on network links as passive elements, prevailing until now. The main conundrum for the unification of these models was a mesh topology, because all of the earlier considered computational paradigms require a fixed number of input links, so they cannot be applied to a mesh.

The origin of this issue is that all the models, except for LGA/LBM, are functional, because they are based on functions with a single output value. Unlike other models, LGA/LBM models are relational, because they are based on relations, which are morphisms of a set of input values onto a corresponding range of output values. Relations in these models are based on equations describing the motion of fluids or gases. In order to make the models computationally tractable, these equations were calculated for a fixed number of inputs and outputs, and the results were implemented as lookup tables. Although the attempts to enhance the geometrical flexibility of LBM continue, they do not have sufficient universality (Ubertini and Succi, 2008). LGA/LBM approach seemed to be very promising, but a more universal, not restricted to a specific underlying grid or physical reality, transformation method was needed. Further investigations eventually led to a new transformation technique and a based on it model, which can be described as a relationist view on interacting systems.

Relationism is a venerable paradigm which can be traced back to Newton's great rival Leibniz, who argued that all properties arise from relations and reality consists of an evolving network of relationships. Later this idea was formulated by Einstein as Mach's principle. In modern physics this approach is revived by Lee Smolin, one of the leading proponents of loop quantum gravity (Smolin, 1997). This approach is also in line with the views of Gregory Bateson, a British anthropologist and cybernetician, who emphasized that logic and quantity are inappropriate devices for describing organisms and their interactions (Bateson, 1969):

"It is impossible, in principle, to explain any pattern by invoking a single quantity. But note that a ratio between two quantities is already the beginning of pattern. In other words, quantity and pattern are of different logical type and do not readily fit together in the same thinking. (p.53) ... We use the same words to talk about logical sequences and about sequences of cause and effect... But the if-then of logic in the syllogism is very different from the if-then of cause and effect... The if-then of causality contains time, but the if-then of logic is timeless. It follows that logic is incomplete model of causality (pp.56-59)".

The difference between a system of causal entailments (what is happening in the external world) and a system of inferential ones (a language in which these events are expressed) was also underlined by the founder of relational biology Robert Rosen (Rosen, 1991). He defined the relation between them via further semantic elements: encoding and decoding, that bring the two entailment structures into congruence (Fig.1c).

Rosen's modeling relation is very close to a cybernetic perception-action loop (Fig.1d), which has become a conceptual schema of a general system, also known as inputprocess-output plus storage (IPO+S) model. From this point of view, all considered models can be divided into two main groups: functional models (CA/RBN/CML) (Fig.1a) and relational models (LGA/LBM) (Fig.1b). The main difference between them is that in functional models a new state of a cell is stored and relayed (fanned out) to all or some of its neighbors, so they can be conservative only in special cases (Fredkin and Toffoli, 1982). In contrast, relational models were created for modeling real physical phenomena, so they are conservative by default. Relational models treat a cell as a "black box" and its responses are only observable to its neighbors but not its internal state. So it can be viewed as a reflexive system differentially responding to its links. Its response (observable reflex) to a link depends on but is not equal to its current state and inputs from other links.

It should be noted that the range of functional models is not restricted to the listed above. All kinds of artificial neural networks, e.g. Spiking Neural Networks (Maass, 1997), and many other dynamical networks with continuous states, e.g. Compositional Pattern Producing Networks (Stanley, 2007), are also functional, because their nodes produce a single output value which is relayed only to some of their neighbors, thus they are not conservative and reflexive.

Rosen's modeling relation, cybernetic and lattice gas models were taken as a blueprint for the kinetic automaton model (Fig.1e), which inherits many of their structural and semantic elements.

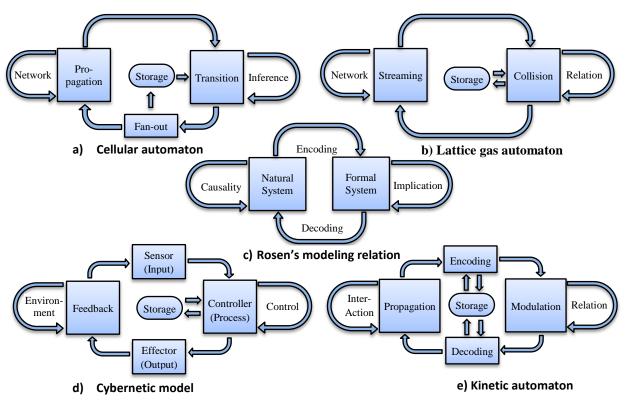


Figure 1: Modeling frameworks

Method

The key element of the model, making its properties and dynamics different from LBM, is a collision step, which is transformed into a 3-step operator: Encoding-Modulation-Decoding, called *Conservative Rank Transform* (CRT). In this method, not quantities as such but their relative values (ranks) are transformed (modulated), and the total quantity does not change after transformation.

CRT is derived from the simplest of all image processing techniques called Intensity Transformations (Gonzalez and Woods, 2008). The main goal of intensity transformation is to increase intensities of some pixels in an image and/or to decrease intensities of others. This is done via an intensity transformation function, which is a mapping of an input value of a pixel onto its output value (Fig. 2).



Figure 2: Basic intensity transformation functions

There is an advanced image processing technique, called Rank Transformation (Zabih and Woodfill, 1993), which changes the intensity of a pixel in relation to its neighborhood. CRT can be outlined as a quantity conserving synthesis of Intensity and Rank Transformations and is carried out in 3 steps:

- **1. Encoding.** The first part of this step, called gathering, is just adding the values of all inputs and storage. The second one, scaling, is the conversion of absolute input values into ratios with a unit sum. In the simplest case it can be ordinary weighting, but in general, it may be a more sophisticated procedure. It should be emphasized that output values must be in the [0,1] range with a unit sum, so they can be regarded as density ratios, probabilities or ranks.
- **2. Modulation.** This is the core step of the method congruent to a collision step in LBM. It modulates (maps) input ratios onto their output values via a kinetic function (map), which is a *normalized* version of the intensity transformation functions in image processing. Modulation of

input ratios is performed on a one-by-one basis, so in a general case, the sum of modulated ratios changes after this step. But this does not contradict to the name of the method. Modulation only changes the proportions between input values. A kinetic map can be any smooth or piecewise curve whose domain and range are defined in the [0,1] interval.

3. Decoding. This step consists of the rescaling of modulated ratios to the volume of storage and scattering them among corresponding outputs and storage.

To illustrate how the method works, a simple numerical example is considered in Figure 3. During an encoding step, a set of input values $\{0.2, 0.4, 1.4\}$, denoted by I, is weighted to the sum of the set. The weights can be regarded as ranks with a unit sum and are denoted by R. On a modulation step, the ranks are modulated via a parabolic kinetic map. The sum of the modulated ranks, denoted by R', increases from 1.0 to 1.6. During a decoding step for obtaining output values, denoted by O, each modulated rank is multiplied by a rescaling factor which is the ratio of the sum of inputs to the sum of modulated ranks (2.0/1.6=1.72). Resultant outputs have the same sum as inputs but different values.

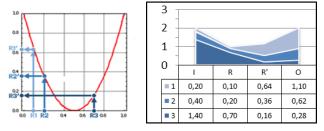


Figure 3: Conservative Rank Transform (CRT)

In a nutshell, this method is a quantity conserving relation or a morphism of a set of input values onto a range of corresponding output values.

Model

The overall schema of the kinetic automaton in detail looks as follows (Fig.4):

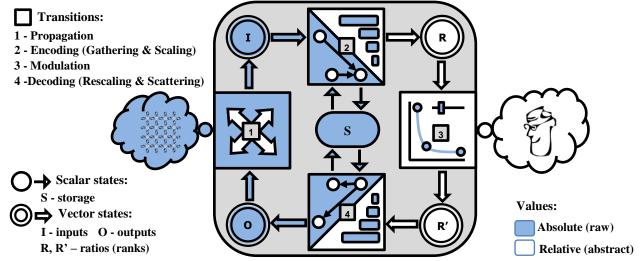


Figure 4: The kinetic automaton state-transition structure

This diagram uses the graphical notation of state-transition structures introduced by Carl Adam Petri and known as Petri nets (Petri, 1982). States in the kinetic automaton model can be thought of as internal buffers for storing intermediate results and are depicted by round shapes, while transitions have square forms. Input (I) and output (O) state buffers of a kinon are represented by real valued vectors of dimension N, where N is the number of neighbors which is fixed for structured and variable for unstructured networks. buffers (R and R') have an additional dimension corresponding to storage. The only buffer in a kinon having a scalar value is storage. It stores the overall quantity during modulation and a quantity remaining in a kinon (its state) during propagation. The sum of output storage values of all kinons of the network can be related to the internal (potential) energy of the network and the sum of output buffers of all kinons to its external (kinetic) energy. The total energy in the network is always the same, whereas internal and external energies interchangeably fluctuate.

In the diagram, a propagation block, operating with absolute values, and storage, holding an absolute (raw) value, are painted in blue (fluid) color. A modulation block operates with relative (abstract) values, so it is painted in white. Encoding and decoding blocks are proxies which manipulate with both types of values, so they are divided into two subblocks and painted in different colors.

In Rosen's modeling relation (Rosen, 1991) (Fig.1c), the relation of equivalence is the following:

$$1 = 2 + 3 + 4$$

where: 1-Natural system, 2-Encoding, 3-Formal system, 4-Decoding.

If we denote divided blocks in Fig.4 as 2N, 2F, 4N, 4F and storage as S, then the following relation of equivalence between the absolute (Natural) and the relational (Formal) parts of the model holds:

$$1 + 2N + S + 4N = 2F + 3 + 4F$$
.

This relation corresponds to Zuse's net automaton, where the left part designates branching lines responsible both for transmission and storage, and the right part is related to the nodes responsible for only information processing. It indicates *the dualism* of the model, because it represents both a cellular and a net automaton in Zuse's terms.

It is easy to notice the similarity of the kinetic cycle to the Carnot cycle and the engines performing it (Fig. 5).

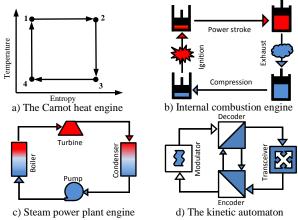


Figure 5: The Carnot cycle engines

This is not a coincidence, because the model was elaborated with having the Carnot cycle as well as Kauffman's workconstraint cycle (Kauffman, 2000) in mind. The workconstraint cycle links the ideas of 'work' and 'constraint', defining work as "the constrained release of energy into relatively few degrees of freedom". A system performs the work-constraint cycle if it is able to use its work to regenerate at least some of the constraints that make work possible. In Kauffman's words, "Work begets constraints beget work". The kinetic automaton was conceived as an abstract autonomous agent doing its own thermodynamical cycle. Work in it is related to the kinetic exchange among automata (interaction) that generates new constraints (relation) inside automata, which in their turn, generate new interaction. Hence, the work-constraint cycle can be put as the interactionrelation cycle: interaction begets relation begets interaction.

The Carnot heat-engine cycle is totally reversible. Therefore, all the processes that comprise it can be reversed, in which case it becomes the Carnot refrigeration cycle. The similarity between the kinetic automaton and the Carnot cycle may lead to the conclusion that the kinetic automaton is also reversible, but it is not true. As it was mentioned earlier, the staple feature of the CRT method is its invariance to the number of inputs and outputs achieved due to decoding. Before scattering, we need to rescale the sum of modulated ratios to the volume of storage, which is an irreversible operation. In Boolean algebra, where states are binary and the number of different combinations is finite, reversibility is possible in special cases (Toffoli, 1980). In real valued algebra it is impossible in principle to split the sum without prior knowledge of components or their ratios, because the number of possible combinations giving the same result is infinite even for two components.

Therefore, this method is in line with both the conservation law and the second law of thermodynamics, which is held to be accountable for the irreversibility of time. It sounds discouraging for the model supposed to generate dynamical patterns. However, despite the second law, Life itself, which is the most improbable and fascinating of all dynamical patterns in the Universe, exists due to the phenomenon of self-organization or "order for free" (Kauffman, 1995). It will be shown further that in most cases the dynamics of kinetic automata networks converges to the total equilibrium or exhibits chaotic behaviour, but in some cases, even subtle changes of the kinetic map can lead to the appearance of stable or periodic patterns from an almost homogeneous state.

The kinetic automaton model can be regarded as a generalized Lattice Boltzmann model, which is not restricted to the Boltzmann equation and a regular grid, but the differences are more than that. Contrary to all considered above models, not quantities as such but their relative values are transformed, which makes the model *relational* in both Rosen's *semantic* and Mach's *relativistic* sense.

Results

The topological universality of the model allows any network structure, but for the ease of the visualization of generated patterns, the Cartesian grid was chosen as the underlying network. It makes the conversion of the system's current state to an image quite straightforward and computationally efficient.

The elementary one-dimensional case will be considered first for better understanding of the basic principles governing the dynamics of the model. A one-dimensional kinetic network consists of N kinons connected in a ring. In all experiments shown below, the total quantity available in the network is equal to N/2. This is equivalent to the average value 0.5 in a uniform state, which is visualized as a grey color in a greyscale image. Corresponding kinetic maps are shown in the left of the images. All shown kinetic maps are defined by splines with a variable number of control points and interpolation order and both axes have [0,1] range.

It is obvious that an identity map always produces a dynamical steady state where nothing changes. However, it seems counterintuitive that an identity map is only one of many others with the same behaviour (Fig. 6a).

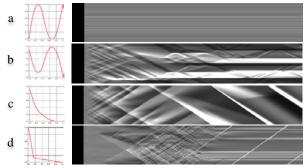


Figure 6: 1D patterns triggered by near equilibrium states

Generally, the dynamics of the model slowly converges to the total equilibrium or produces chaotic behaviour. However, some kinetic maps, in accordance with Turing's idea, do generate a non-uniform steady state (Fig. 6b) or give rise to travelling waves (Fig. 6c) behaving like solitons and only slightly changing after collisions. Even more, generated standing and travelling waves can coexist (Fig. 6d).

Now we change initial settings to the opposite extreme, where only one kinon has a storage which is equal to the total quantity of the network, so it can be termed as a singularity. Remarkably, the dynamics remains almost the same (Fig.7).

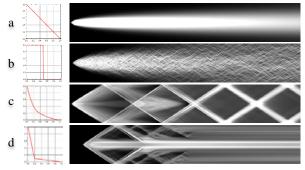


Figure 7: 1D patterns triggered by singularity

It is not surprising that configurations, where the storage of all kinons in the network is randomly set in the [0,1] range, behave similarly. Except for a totally uniform state, which cannot be changed by any kinetic map, the dynamics can be very sensitive to the tuning of the kinetic map in some cases and almost indifferent in others. It means that under certain

conditions there exists a phase transition, which typically refers to a very narrow transition domain from one macroscopic state of the system to another. It is characterized by a sudden change in some order parameter $\varphi(\mu)$ that depends on some control parameter μ (e.g., temperature, density, probability, etc.) that can be continuously varied (Solé, 2011). Although relevant order and control parameters of the model are yet to be identified and studied, but experiments with a very simple kinetic map shown in Figure 8 revealed the existence of a narrow range of parameter k when a nearly uniform initial configuration converges to a dynamical ordered or chaotic pattern (II: 1.5 < k < 2) rather than a uniform (I: k<1.5) or non-uniform stable (III: k>2) state. Picture in the bottom of Figure 8 shows the change of dynamics under periodic increases of parameter k. The most dramatic change occurs at k=1.6 when a nearly uniform state triggers spontainiosly into an ordered periodic pattern.

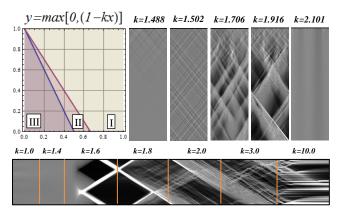


Figure 8: One-dimensional phase transitions

The long history of cellular automata research demonstrates that it is very difficult to produce round shapes on a square grid, but in kinetic automata it goes without any effort. In the simplest 2D configuration with a singularity and a negative kinetic map after several initial steps, an expanding wave quickly changes its form from a diamond to an ideal circle. The boundary slowly dissolves into a halo and the system converges to the total equilibrium in the long run (Fig. 9)



Figure 9: Kinetic "Big Bang"

It was shown in Figure 8 that some changes of a kinetic map can induce a sharp phase transition. If we rerun the previous experiment and, after about 150 steps, when a full-fledged circle is formed, change the kinetic map parameter k=1.0 for $k\sim4.0$, an expanding circular wave splits into four sectors which begin travelling in opposite diagonal directions (Fig. 10). In all shown experiments, a kinetic map is the same for all kinons in the network and is changed manually, but there are many ways to subject kinetic map parameters to the current state of a kinon and/or its inputs, which can make its dynamics even more complicated and unpredictable.

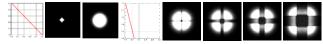


Figure 10: Kinetic fission

If we choose a parabolic kinetic map shown in Figure 3 and start from a singularity again, then fascinating kaleidoscopic ornaments will change non-repetitively for a long time. The picture will be even more fascinating with four evenly spaced singularities (Fig. 11).

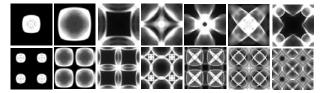


Figure 11: Kinetic kaleidoscope

Some kinetic maps can produce dynamic patterns which converge to a stable non-uniform state in conformity with the Turing two-factor sytems, but unlike Turing patterns, these states can be very intricate (Fig. 12):

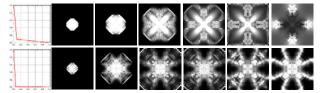


Figure 12: Kinetic "Still Life"

In most case studies starting from a nearly homogeneous state, a uniform grey square is usually observed, but now and then the uniformity breaks out in spots, which then turn to the chaotic or non-uniform stable state (Fig. 13)

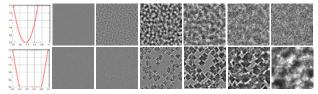


Figure 13: Chaotic and stable kinetic patterns

It seems that it is more than enough for such a relatively simple model and it is highly unlikely to expect more, but in fact, its creativity is endless. In rare cases, one can observe almost improbable life-like phenomena, which can move, grow, merge, split and dissolve like Conway's Game of Life dynamical patterns: gliders, puffer trains, avalanches, etc. (Fig. 14)

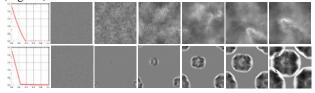


Figure 14: Kinetic Game of Life

Discussion

These examples, which are an infinitely small portion of all possible patterns, prove the main thesis of this paper that even one-component kinetic networks can generate stationary and dynamic patterns similar to multi-component Turing systems. Unlike Turing systems, they are triggered by a single instability, tentatively called *kinetic instability*, which is an

instance of a general *self-organized instability* (Solé et al. 2002). Multi-component kinetic networks are also possible and can be implemented by layering of all values in kinon buffers into vectors of any dimension. Apart from the same or different kinetic maps governing the dynamics of each layer there can be additional coupling among layers ('kinetic chemistry') which can be also implemented via a kinetic map. The state and dynamics of three-component kinetic networks can be readily visualized by color images.

Although the CRT method, described above and used as a computational kernel in all shown experiments, proved to be elegant and expressive, it is only one of many other possible transformation methods. The main requirement of the model is *quantity conservation* during the mapping of multiple input values and storage onto their output (feedback) values.

Closer examination revealed many unusual phenomena not found in Turing systems. In all case studies starting from a singularity, the initial diamond expanding wave, determined by the underlying square grid, quickly converges to a circle. One of the possible explanations can be done with the use of the concept of Brillouin zones. Léon Brillouin, a French physicist who studied wave propagation in periodic structures (Brillouin, 1946), introduced the concept of zones named after him. Physically, Brillouin zone boundaries represent Bragg planes which reflect waves having particular wave vectors, so that they cause constructive interference. The constriction of the first two Brillouin zones on a square grid and the final 10th Brillouin zone is shown in Figure 15.



Figure 15: Brillouin zones

The most enigmatic phenomenon observed is an almost instantaneous fission of the expanded wave into four sectors after the change of the kinetic map (Fig.10). There is no acceptable explanation for it so far, but without a doubt, it is a self-organized phenomenon, because it requires a large-scale correlation.

Some generated patterns (Fig.11 and 12) demonstrate striking similarities with real-world phenomena, e.g., periodic patterns obtained by a Swiss medical doctor and natural scientist Hans Jenny, who coined the term *Cymatics*, the study of wave phenomena and vibrations (Jenny, 2001). Jenny pioneered the use of piezoelectric crystals hooked up to amplifiers and frequency generators to make the resultant nodal fields visible by spreading a fine powder of lycopodium spores, as well as many other materials (Fig. 16). The shapes, figures and patterns appeared to be primarily a function of frequency, amplitude, inherent properties of the materials and the size of the plate.

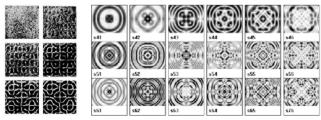


Figure 16: Cymatic patterns

Conclusion

The presented results convincingly show the fidelity of the model and demonstrate high potential for future research. According to Stephen Wolfram, "In continuous cellular automata it takes only extremely simple rules to generate behaviour of considerable complexity" (Wolfram, 2002). It was shown here that relational approach dramatically increases the complexity of the behaviour of the kinetic automaton model. Remarkably, the proposed model structurally coincides with the recently suggested Bayesian model of perception (Friston et al., 2012), although these models are based on different assumptions and approaches. This confirms the universality and wide applicability of the model. It was demonstrated that kinetic automata possess innate tunability, which makes them a candidate toy model and a playground for studies in self-organization in general and guided self-organization (Prokopenko, 2014) in particular. The analog nature of the model enables its direct implementation with analog circuits, so the advent of memristive technologies (Adamatzky and Chua, 2014) and the revival of analog computing (Mills, 2008) may lead the way for kinetic automata to tunable kinetic media.

References

- Adamatzky, A. and Chua, L. editors (2014). *Memristor Networks*. Springer
- Bateson, G. (1979). *Mind and Nature: A Necessary Unity*. New York: E. P. Dutton.
- Brillouin, L. (1946). Wave Propagation in Periodic Structures: Electric Filters and Crystal Lattices. McGraw-Hill
- Cariani, P. (1993). To Evolve an Ear: Epistemological Implications of Gordon Pask's Electrochemical Devices. Systems Research 10(3):19–33
- Fredkin, E. and Toffoli, T. (1982). Conservative Logic. *Int. Journal of Theoretical Physics* 21 (3-4): 219–253
- Frisch, U., Hasslacher, B., Pomeau, Y. (1986). Lattice-gas automata for the Navier-Stokes equations. *Phys.Rev. Lett.* 56, 1505-1508
- Friston, K., Breakspear, M., Deco, G. (2012). Perception and selforganized instability. Front. Comput. Neurosci. 6: 44.
- Gardner, M. (1970). The Fantastic Combinations of John Conway's New Solitaire Game "Life". Scientific American 223:120-123
- Gonzalez, R.C. and Woods, R.E. (2008). *Digital Image Processing*. 3rd Edition. Prentice Hall
- Jenny, H. (2001). Cymatics: A Study of Wave Phenomena & Vibration. Vol. I & II (3rd ed.). Macromedia Press
- Kaneko, K. (1985). Spatiotemporal Intermittency in Coupled Map Lattices. Prog. Theor. Phys. 74, 1033-1044
- Kauffman, S.A. (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22:437–467
- Kauffman, S.A. (1993). The Origins of Order. Oxford University Press
- Kauffman, S.A. (1995). At Home in the Universe: The Search for Laws of Self-Organization and Complexity. Oxford University Press
- Kauffman, S.A. (2000). Investigations. Oxford University Press
- Klafter, J. and Sokolov, I.M. (2005). Anomalous diffusion spreads its wings. *Physics World*, 18(8): 29–32.
- Maass, W. (1997) Networks of spiking neurons: the third generation of neural network models. *Neural networks* 10 (9): 1659-1671

- Meinhardt, H. (2004). Out-of-phase oscillations and traveling waves with unusual properties: the use of three-component systems in biology. *Physica D*, 199:264-277
- Mills, J. (2008). The nature of the Extended Analog Computer. *Physica D*, 237:1235-1256
- Murray, J. D. (1993). Mathematical Biology. Berlin: Springer Verlag.
 Neumann, J.V. (1960). Theory of Self-Reproducing Automata. Burks,
 A.W. editor. Univ. of Illinois Press
- Pask, G. (1961). A Proposed Evolutionary Model. In Foerster von, H. and Zopf, G. editors, *Principles of Self-Organisation*. Pergamon Press, 229–254
- Petri, C.A. (1982). State-Transition Structures in Physics and in Computation. *Int. Journal of Theoretical Physics*, Vol. 21, No. 12, pp. 979-992
- Pour-El, M.B. and Richards, J.I. (1989). Computability in Analysis and Physics. Berlin: Springer-Verlag
- Prokopenko, M. editor (2014). Guided self-organization: Inception. Springer Series on Emergence, Complexity and Computation, Vol.9
- Rosen, R. (1991). *Life Itself: A Comprehensive Inquiry into the Nature, Origin, and Fabrication of Life.* New York: Columbia University Press.
- Smolin, L. (1997). The Life of the Cosmos. Oxford University Press
- Solé, R.V., Alonso, D., McKane, A. (2002). Self-organized instability in complex ecosystems. *Philos. Trans. R. Soc. Lond. B Biol.* Sci. 357(1421):667-71.
- Solé, R.V., (2011). Phase Transitions. Princeton University Press: Princeton, NJ
- Stanley, K. (2007). Compositional Pattern Producing Networks: A Novel Abstraction of Development. Genetic Programming and Evolvable Machines. 8 (2):131–162.
- Toffoli, T. (1980). Reversible computing. Technical Report MIT/LCS/TM No. 151, MIT
- Turing, A.M. (1952). The chemical basis of morphogenesis. *Phil. Trans. of the Royal Society of London, B* 237:37–72
- Ubertini, S. and Succi, S. (2008). A generalised lattice Boltzmann equation on unstructured grids. *Comm.Comp.Phys.* 3, 342-356
- Ulam, S. (1952). Random processes and transformation. Proc. International Congress of Mathematicians. Cambridge, MA, August 30-September 6, 1950), vol. 2: 264–75
- Walgraef, D. (1997). Spatio-Temporal Pattern Formation, with Examples in Physics, Chemistry and Materials Science. Springer, New York.
- Wolf-Gladrow, D.A. (2000). Lattice-gas cellular automata and lattice Boltzmann models: an introduction. Springer, Berlin
- Wolfram, S. (1986). Cellular Automaton Fluids 1: Basic Theory. Journal of Statistical Physics, Vol.45(3/4): 471-526.
- Wolfram, S. (2002). A New Kind of Science. N.Y.: Wolfram Media Zhabotinsky, A.M, Dolnik, M., Epstein, I.R. (1995). Pattern formation arising from wave instability in a simple reactiondiffusion system. Journal of Chemical Physics 103:10306.
- Zabih, R. and Woodfill, J. (1994). Non-parametric local transforms for computing visual correspondence. Eklundh J.-O., editor, Computer Vision–ECCV '94, v.801 of LNCS, Springer, Berlin, pp.151–158.
- Zuse, K. (1969). Rechnender Raum. Braunschweig: Friedrich Vieweg & Sohn [English translation: Calculating Space. MIT Technical Translation AZT-70-164-GEMIT, MIT (Proj. MAC), Cambridge, Mass. 02139, Feb. 1970]
- Zuse, K. (1975). Ansätze einer Theorie des Netzautomaten. Nova acta Leopoldina, Leipzig

Evolution of communication-based collaborative behavior in homogeneous robots

Onofrio Gigliotta¹ and Marco Mirolli²

¹Natural and Artificial Cognition Lab, University of Naples Federico II, Napoli, Italy ²Institute of Cognitive Sciences and Technologies, CNR, Rome, Italy onofrio.gigliotta@unina.it, marco.mirolli@istc.cnr.it

Abstract

In the field of collective robotics much research has been devoted to the study of coordinated and cooperative behaviours where typically all the robots play the same function. Much less attention has been devoted to the development of groups of robots that play different roles (robot teams), probably because evolutionary collective robotics tend to use groups of homogeneous robots, in which role differentiation poses difficult challenges. In the few Evolutionary robotics studies in which role differentiation has been demonstrated such differentiation depends exclusively on the robots physical interactions, making the solutions found by evolution quite fragile, in particular with respect to the number of robots that form the group. In this paper we apply a method for role differentiation developed in previous work to the evolution of teams of homogeneous robots in which role differentiation is based on a dedicated communication channel. Our evolved robots are able to negotiate their role through communication and perform very effectively their collaborative task, which requires that one robot is sent to a 'mission' away from the group while all other robots remain in a 'home'. Our simulations also show that the method proposed, based on the rewarding of communication-based role differentiation, is necessary for the evolution of the desired behaviour. Finally, we show that since role differentiation is based on communication and not only on robot physical interactions, evolved solutions are considerably robust with respect to the number of robots composing the group.

Introduction

Recently the study of collective robotics has been raising increasing interest in the Artificial Life community. In particular, Evolutionary Robotics techniques (Nolfi and Floreano, 2000; Harvey et al., 2005) have proven extremely successful to develop collective behaviours in group of robots (see, e.g. Baldassarre et al., 2003; Dorigo et al., 2004; Spector et al., 2005; Quinn et al., 2003; Brambilla et al., 2013). Evolutionary Robotics is a design methodology in which populations of robots are evolved for a specific task, defined by a *fitness function*, for a number of generations, until a satisfying solution to the task is found. The parameters that determine the behaviour of the robots (for example the connection weights of their neural networks) are encoded in their

genomes and are selected on the basis only of the overall performance of the robots, defined by the fitness function. In this way, the evolutionary process can exploit the full power of self-organisation in which behaviours emerge from the interactions between the robots and their environment and, in the case of collective robotics, among the robots themselves (Nolfi, 1998).

According to Garnier et al. (2007) collective behaviours can be categorised in four kinds: 1) coordinated behaviours, where the group produces a specific spatio-temporal organization of the relative positions of its members and/or of the results of their activities which is functional with respect to a certain goal; 2) cooperative behaviours, where individuals combine their efforts to solve a problem that no one individual can solve by itself; 3) collective decision making, where a group of individuals collectively chooses one among several opportunities in order to maximize the performance with respect to a given problem; 4) collaborative behaviours, where a collective goal is achieved through different activities that are simultaneously performed by different individuals.

Most of the work that has used evolutionary robotics methods for developing collective behaviours has been devoted to coordinated or cooperative behaviours, in which the individuals do not need to differentiate their roles. This is probably due to the fact that evolutionary collective robotics typically works with groups of homogeneous robots, that is groups of robots with identical genomes and hence identical control systems. The reason is that interesting collective tasks tend to require cooperativeness and if the interacting agents are non-homogeneous the emergence of cooperation is extremely difficult due to the problem of altruism (Mirolli and Parisi, 2005 and Floreano et al., 2007 are two examples of works dedicated to the problem of altruism in groups of communicating agents). On the other hand, the use of homogenous robots makes it difficult the development of robot specialisation.

In this paper we are interested in the development of collective behaviours, where interactive robots form a *team*: i.e. (1) different individuals make different contributions to the

task, (2) cooperation is required as roles are interdependent, and (3) the group organisation persists over time (see the definition of team provided by Anderson and Frank, 2001 in the context of animal behavior).

Related work

A few previous works have been devoted to the evolution of role differentiation in groups of robots (Baldassarre et al., 2003; Quinn et al., 2003; Tuci et al., 2013). Baldassarre et al. (2003) evolved a group of robots for the ability to collectively navigate toward a light target. Beyond proximity and light sensors, each robot had a speaker producing a constant fixed sound and directional microphones for detecting the sound produced by the other robots. Evolved robots displayed different kinds of strategies, the most effective of which involved role specialisation: in particular, one individual tended to assume the frontal position and drive the others towards the light while the others stayed in the back and followed the 'leader'. Since the robots were homogeneous and their controllers were single layer perceptrons without any internal state, robot specialisation was completely situated, that is completely dependent on the different sensory stimulation that different robots received from the environment.

In a similar work, Quinn et al. (2003) evolved a team of three homogeneous robots for the ability to move together in the environment. The peculiarity of this experiment was the minimalism of robot equipment: each robot had just four infrared sensors and two motor-driven wheels. Evolved robots were able to navigate together by relying on a strategy that had two phases: in the first phase the robots negotiated their roles until they reached a line formation; in the second phase they started to move by swinging clockwise and anti-clockwise while maintaining their relative positions.

Finally, Tuci et al. (2013) evolved a group of homogeneous robots for their ability to perform two behaviours: some robots had to remain and patrol the 'nest', while other robots had to leave the nest and go foraging in a food area. Furthermore, the role allocation strategy depended on the context: in one context the number of robot patrolling the nest had to be higher than the number of robots foraging, while in a second context the opposite holded.

Notwithstanding the importance of these works, they developed solutions that are not general but task-specific: the dynamic role allocation demonstrated by these robots can be used only for the tasks for which the robots were evolved and cannot be exploited for any other purposes. Furthermore, those solution do not seem to have the robustness that is typically assured by the use of homogeneous robots. As correctly discussed by Quinn and colleagues, the use of homogeneous robots has the potential to assure high robustness: since all robots are identical and hence equally able to play any role, teams of homogeneous robots should be in principle able to cope with the loss of individual members

or even to the addition of more robots. However, this advantage is not demonstrated in any of the works discussed above: Baldassarre et al. did not touch the problem of robustness, and it is not clear whether their robots' flocking behaviour would generalise with respect to the number of robots in the group; Quinn et al. report that their robots were not able to cope with the lack of one individual; Tuci et al. report that their evolved strategies "seem quite fragile with respect to various sources of variability, such as the cardinality of the team...". A possible reason for this fragility is that the evolution of task-specific role allocation will tend to produce task-specific solutions which deeply rely on the specific conditions under which evolution took place.

In order to solve this problem, in previous work (Gigliotta et al., 2009) we have proposed a simple mechanism for evolving dynamic role allocation in groups of robots that is based on a dedicated communication channel. In particular, robots were evolved for producing different communication signals: for example, one task consisted in having one robot (the 'leader') producing one signal and all other robots (the 'followers') producing another one (another task was to have half of the robots producing one signals and the other half the other signal). We found that evolved solutions not only involved the co-adaptation of communicative and noncommunicative behaviours (Nolfi, 2005; Mirolli and Nolfi, 2010), but, most importantly, that they were very robust with respect to the number of robots involved. However, the limit of that work was that role-allocation was the only behaviour that the robots were required to show: in other words, fitness depended only on the signals emitted by the robot and role differentiation had no other adaptive function.

In this paper we apply the idea of communication-based dynamic role allocation to a simple task in which one robot (the 'leader') has to leave the group to perform its own task, while all other robots (the 'followers') have to remain together in their home area. We compare a condition in which the fitness function rewards only the noncommunicative behaviour with one in which we add to this fitness function a component, inherited by our previous work, which also rewards communicative role-allocation. In this way we demonstrate that our idea of communicationbased role-allocation can indeed be exploited to develop non-communicative team behaviour in homogeneous robots. Furthermore, since this kind of role allocation does not depend on the task nor on the number of robots, we also show that the resulting behaviour is robust with respect to the number of robots constituting the team.

Experimental set-up

Robot and task

We want to demonstrate that a group of homogeneous robots can dynamically and autonomously learn to allocate different roles (i.e. 'leader' and 'followers') and behave according to these roles. Furthermore, we want to demonstrate that the

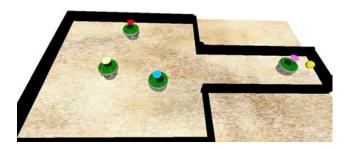


Figure 1: The set-up

robot form a real team, where the group organisation persists over time (see above) and is not strictly dependent on the continuous physical interactions between the robots. For these reasons the metaphor that guided us for developing our experimental set-up was that of a group of robots which had to send one and only one robot away from the group in order to accomplish a 'mission' (e.g. finding resources, exploring some other place, rescuing somebody...).

The experimental set-up (fig. 1) consists in four simulated e-puck robots placed in an arena formed by a 'home' square (side = 600 mm) and a rectangular corridor (500 x 200 mm). At the end of the corridor there is a small light (yellow sphere in fig 1). The task for our group of robots is to send one of the robots into the corridor to get close to the light while all other robots remain in the home area.

Neural controller

The controller of the robots is the neural network depicted in fig 2. The sensory system is composed by: 8 infrared sensors, placed around the robot's body, which provide a noisy indication of the proximity of an object (another robot or a wall): 8 light sensors, also placed around the robot's body. which provide a noisy indication of the proximity of the light (if within sight, i.e. at about 300 mm); one communication sensor, which encodes the value of the highest signal emitted by the other robots (within a distance of 1000 mm, meaning that basically each robot can hear each other within the arena); and a unit which encodes the signal emitted by the same robot in the previous time step (for the importance of talking-to-oneself in artificial life and robotics, see Mirolli and Parisi, 2005, 2011). All sensory units are directly connected both to the output units and to a group of 8 fully recurrent internal units, which in turn send connections to the output units. The group of output units is composed by the two motor units, encoding the speed of the two wheels, and one communication unit, which encodes the signal emitted by the robot (in [0, 1]). Output units are simple logistic units, whose activation o_i is given by

$$o_i = \phi(I_i), \phi(x) = (1 + e^x)^{-1}, I_i = \sum_{j=1}^{N} a_j \times w_{ij} - b_i$$
 (1)

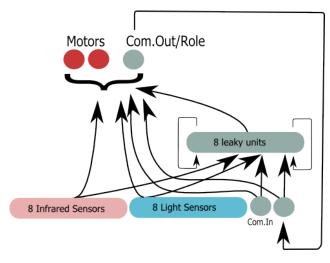


Figure 2: The neural controller

where a_j is the activation of the unit j that sends connection w_{ij} to output i and b_i is the bias of unit i. The activation of each motor unit (in [0,1]) is remapped in [-8,+8] and is sent to the motors of the robot. Internal units are leaky integrator units, whose activation is given by

$$o_i^t = (1 - \tau_i) \times \phi(I_i^t) + \tau_i \times o_i^{t-1}$$
(2)

where τ_i is the time constant (in [0,1]) of unit i.

Evolutionary algorithm

The genome of a robot encodes the values of the connection weights, the biases and the time constants of the internal units. Each parameter is encoded as an 8 bits string, whose value is uniformly mapped in the range [-5.0, +5.0]for weights and biases and in the range [0,1] for time constants. The first population consists of 100 randomly generated genomes. Each genome is tested for 15 trials, each lasting 2000 time steps. At the beginning of each trial the genome is translated in the corresponding neural controller, and the same controller is embedded in each of the four robots that constitute the group. The four robots are randomly placed into the home square of the arena and let free to move and communicate for the rest of the trail. After the fitness of each genome in a generation has been calculated (see below), the 20 best genomes are copied 5 times each (reproduction) and 1% of the bits of each new genome is flipped (mutation). The evolutionary process lasts 1000 generations.

We run two kinds of experiments which differ only for the fitness function employed. In the first experiment, which we will call 'base-line', the fitness function is purely behavioural, rewarding only the ability of the robots to stay in the configuration that we are aiming for, that is having one and only one robot in the corridor close to the light while all other robots remain in the home square. In particular, the fitness of the base-line experiment is composed by two components, rewarding separately a) the behaviour of the 'leader' (defined as the robot who is, in each time step, most close to the light), and b) the behaviour of the 'followers' (i.e. all other robots). In particular, the first behavioural fitness component (BFC1) measures how close the leader is to the light, normalised in [0, 1]:

$$BFC1(g) = \frac{\sum_{t}^{T} \frac{max(0, (M - d(L, light)))}{M}}{T}$$
(3)

where M is a maximal distance, set to 900 mm and d(L, light) is the distance between the leader and the light, and T are all the time steps of all the trials of a genome's 'life' (i.e. 1000 time steps x 10 trials = 10000).

The second component (BFC2) measures the average distance of the followers from the light, again normalised in [0,1]:

$$BFC2(g) = \frac{\sum_{t}^{T} \sum_{i}^{F} \frac{min(1, d(F_{i}, light))}{M}}{T \times F}$$
(4)

where F_i is follower i and F is the number of followers, i.e. 3. Since the behaviour of the leader is more important than that of the follower, the global behavioural fitness (BF) gives more weight to F1 than to F2, in particular:

$$BF(g) = 0.75 \times BFC1(g) + 0.25 \times BFC2(g)$$
 (5)

The second experiment, which we will call the 'communication rewarded' experiment is identical to the first one (the ratio between the leader and the followers fitness components is kept constant) but for the addiction of another component to the fitness, which is identical to the fitness that we used in our previous work (Gigliotta et al., 2009) for evolving role-allocation based on communication. In particular, this component rewards the genome for having one robot sending a high value signal and all the other sending a low value signal. More precisely, the communication fitness component (CFC) is measured as follows:

$$CFC(g) = \frac{\sum_{t}^{T} \sum_{i}^{N} O_{max} - O_{i}}{T \times (N-1)}$$
 (6)

where O_{max} is the highest signal value, O_i is the value of the signal of robot i, and N is the number of robots in the group, i.e. 4. Finally, the global fitness of the communication rewarded simulation is

$$CRF(g) = 0.8 \times BF(g) + 0.2 \times CFC(g) \quad (7)$$

$$= 0.2 \times BFC1(q) + 0.6 \times BFC2(q) + 0.2 \times CFC(q)$$
 (8)

Each experiment is replicated 20 times by starting with different random conditions.

Results

Figure 3 shows the evolution of the fitness of the best individual of each generation for the two experiments: both the global value and the three different components are shown (average results of 20 replications). By looking at the global fitness (that determining the evolutionary process: fig 3a), we can see that the communication rewarded condition is much better than the base-line one: in the former we can clearly see an evolutionary improvement (from 0.3 to 0.75); in the latter fitness is almost flat (from 0.3 to 0.4). It is already clear that this difference can not be explained only by the communicative fitness component, which influences the fitness of the communication rewarded condition only for one fifth (0.2). Indeed, if we look at the fitness related to the behaviour of the leader (BFC1, fig. 3b) we see that there is a big difference between the two conditions. At the end of the evolution, in the communication rewarded condition the best genomes get on average a score of more than 0.5, meaning that the leaders stay on average at a distance of less than 450 mm form the light, which is inside the corridor, while in the base-line condition they score 0.2. Even more interesting is the evolution of the component related to the behaviour of the followers (BFC2, fig. 3c): while in the communication rewarded condition this component is stable to 0.9, in the base-line condition it decreases during evolution. Such a decrease is concomitant with the increase in the first behavioural fitness component, meaning that as the leader learns to get closer to the light, so do also the followers. Finally, the plot of the communication component, which is present only in the communication rewarded condition, shows that the evolution of this component is very fast: it reaches a steady state of about 0.95 after about 50 generations. In particular, this fitness component evolves well before than the (leader) behavioural component. This, together with the fact that the absence of this component prevents the evolution of the desired behaviour, shows that it is just the ability to dynamically allocate tasks through communication, evolved thanks to the communication fitness component, that allows the emergence of the ability to solve the behavioural task.

This conclusion is further corroborated by looking at the communicative behaviour of the two conditions. In particular, we measured how much the signals of the robots in a group are differentiated by running, for each of the best evolved individuals of the two conditions, 100 test trials. During these tests we recorded, for each timestep, the robot communicative behaviour by binarizing the communication output: we assign 1 to robots whose communication unit exceeds 0.5 and 0 otherwise. Hence, for a group formed by 4 robots we have $16 (2^4)$ possible configurations. The frequencies of such states are reported in fig 4. Furthermore, we computed the entropy (Shannon, 1948) of the two distri-

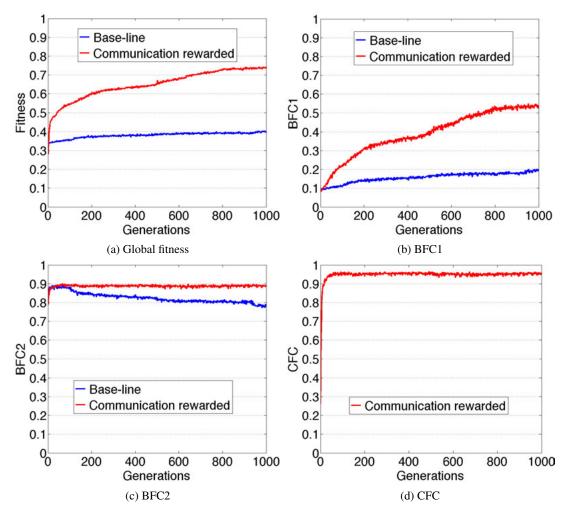


Figure 3: Evolution of fitness. All data refer to the averages of the best individuals of the 20 replications. BFC1 = first behavioural fitness component (related to the leader). BFC2 = second behavioural fitness component (related to the followers). CFC = communication fitness component (only present in the communication-rewarded condition.

butions (of the two conditions):

$$H = -\sum_{i=1}^{16} p_i \log_2 p_i \tag{9}$$

where p_i is the experimental probability (i.e. the relative frequency) of the i_{th} configuration.

While in the communication rewarded condition the signals are differentiated (since, as requested by the communication fitness component, one robot sends a high value signal while the others send a low value one) presenting an entropy equal to 2.23, in the base-line condition the entropy is very close to 0 (0.00036), meaning that all robots send always the same signal (that is 0 in this case). This means that the robots of the base-line condition do not use communication at all. The inability of the base-line condition to evolve an appropriate communication system with which to communicate the different roles to one another is the reason for

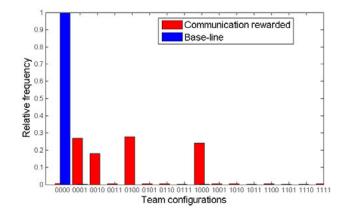


Figure 4: Frequency distribution of team configurations for the best individual of the base-line and the communication rewarded condition.

the inability to evolve the appropriate non-communicative behaviour. Indeed, given that all robots in a group have the same body and the same control system, in order to behave differently they need something that might differentiate the two behaviours of a) going in the corridor and towards the light (leader) and b) remaining outside the corridor in the home (followers). The only thing that seems to be able to permit this differentiation is communication. In fact, the signals have a twofold function. The first is social, that is to make sure that in the group there is one and only one leader. The second is individual, that is to permit the differentiation of individual behaviours: the robot that sends a high signal is the one that looks for the corridor, enters it, and is attracted by the light; the robots that send low signals remain in the home.

The difference between the results of the base-line and the communication rewarded conditions can be best appreciated if we measure the performance of the groups of robots on the basis of the task that we wanted them to accomplish in the first place. Our goal was to have one robot to go 'on mission' inside the corridor and all the others to remain in the home. For this reason we measured performance as the proportion of time steps in which the robot are in the desired configuration, i.e. one robot in the corridor and all the other outside. Furthermore, since it necessarily takes some time for the group to get to the desired state (as roles needs to be negotiated and the leader has to find the corridor and enter it), we take the performance measure only in the last 100 time steps of a trial (choosing a different interval like the second half of the trial or just the last time step gives almost the same results). The average results of the 20 replications of the two conditions in the 100 test trials are shown in fig. 5. The results speak for themselves. The median of the communication rewarded condition is about 0.9, meaning that in most of the replications the robots solve the task almost always; there are also some 0s, as in a few replications the desired behaviour has not evolved (yet?). In striking contrast, the performance for all the best evolved genomes of the base-line condition is 0, meaning that no group of that condition has ever achieved the desired configuration. This result is related to the fact, shown above, that as the fitness component rewarding the behaviour of the leader increases that of the behaviour of the follower decreases. Given that in the base-line condition robots never learn to differentiate their communicative behaviour and hence their roles, in those replications in which the behaviour of entering the corridor and approaching the light evolves, all the robots of the group exhibit that same behaviour, resulting in a configuration where all the four robots queue inside the corridor towards the light. Given that the behaviour of the leader is weighted more, in the fitness, than the average behaviour of the followers (eq. 5), that different robots enter the corridor at different times, and that for physical reasons the last robot to enter the corridor remain distant from the light be-

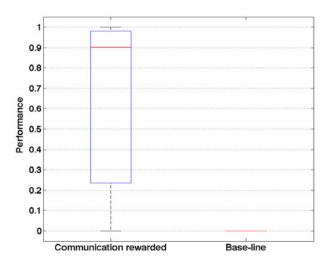


Figure 5: Post-evaluation results for all the 20 best individuals of the base-line and the communication rewarded conditions. Performance is calculated as the proportion of time steps, among the the last 100 of a trial, in which the group of robots accomplishes the task, meaning that one and only one robot is inside the corridor. Each box represents the inter-quartile range of the data, while the red line inside the box marks the median value. The whiskers extend to the most extreme data points within 1.5 times the inter-quartile range from the box.

cause they have other robots in front of them, this kind of behaviour tends to provide a higher fitness than the behaviour of remaining all the robots in the home, but nonetheless, it does not solve the task, and results in a performance of 0.

Finally, we tested the robustness of the best solution to the variation in the number of robots of the group. We took the best individual out of all the replications of the communication rewarded simulations and we tested its performance with a number of robots that goes from 2 to 10. In particular, for each test we run 100 trials where we measure, as before, the proportion of time steps in which one and only one robot is in the corridor, during the last 100 time steps of each trial. The results (fig. 6) show that the evolved solution is extremely robust to the number of robots in the team: from 2 to 7 robots performance is above 0.9, and even with 10 robots we obtain a quite high performance of about 0.65. Furthermore, by observing the behaviour of the team with 10 robots, we can see that even when the robots fail to accomplish the task, this is not due to a disruption of the group behaviour, but rather to the fact when there are so many robots the arena is overcrowded and it becomes difficult for the leader to find the corridor and enter it. If we would just prolong the duration of the trial or enlarge the arena the performance would return to optimal values even with 10 robots.

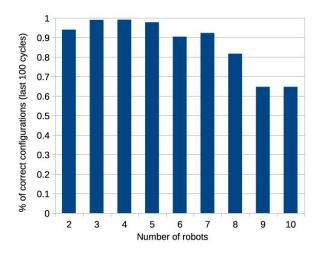


Figure 6: Generalisation tests of the best evolved individual of the 'communication rewarded' condition. The proportion of time steps (among the last 100 of a trial) in which one and only one robot is in the corridor is plotted against the number of robots in the team.

Discussion and conclusion

In this paper we have demonstrated that it is possible to evolve a team of homogeneous robots in which the robots autonomously negotiate different roles among themselves and behave according to the allocated roles. In particular, our robots are able to elect a 'leader' that is sent to a 'mission' outside the home area, while all other 'follower' robots remain in the home.

In contrast to all the analogous previous works which we are aware of, where the evolved behaviours were very fragile, in particular with respect to the number of robots, our system proved to be extremely robust, both to the addiction and to the subtraction of robots in the group. We maintain that this high robustness depends on the presence of a communication channel that is used to allocate the roles. If, as in previous works (Baldassarre et al., 2003; Quinn et al., 2003; Tuci et al., 2013) role allocation depends only on robots behaviour, it will tend to be situated, that is dependent on the context. This means that as the context changes, for example because the groups contains a different number of robots, then the behaviour will tend to stop functioning. On the contrary, if roles are allocated and maintained through a dedicated communication channel, they will tend to depend much less from the context, which renders the system much more robust.

The present work demonstrates that the method we had proposed for communication-based role allocation in previous work (Gigliotta et al., 2009) can be effectively applied to develop robots able to accomplish non-communicative collaborative task. Furthermore, our simulations show also that, in some circumstances (like the ones of our scenario), di-

rectly rewarding communication-based role allocation can be necessary for the non-communicative team behaviour to evolve. The fact that if communication-based role allocation is not rewarded through the fitness function the robots cannot solve the task suggests that it is too difficult to co-evolve the necessary communicative and non-communicative behaviours from scratch. Consider that the evolution of communicative behaviours per se can be difficult as the traits which are necessary for its emergence - namely, sending good signals and responding appropriately to received signals - taken in isolation do not increase the reproductive chances of the individuals that possess them (Maynard-Smith, 1997; Mirolli and Parisi, 2008). Furthermore, our robots have to develop not only their communicative behaviours, but these must be co-adapted to the appropriate non-communicative behaviours, that is going 'on mission' for the leader and remaining in the home for the followers. Even if this is surely possible in principle, our simulation show that it is too difficult in practice. On the contrary, if communication-based role allocation is rewarded through the fitness function the evolution of the desired behaviour becomes feasible, and even easy (remember that the median performance in the tests of the best individuals of 20 replications of the experiments was 0.9). The reason is that, if explicitly rewarded, the evolution of communication-based role allocation is straightforward, and once role are allocated the evolution of the behaviours that are appropriate to the roles is not so difficult.

Some researchers in the Evolutionary Robotics community may not like the explicit rewarding of the communicative behaviour of our robots, as this might sound like 'cheating', in the sense of providing too much information and canalising the evolutionary process. Even though we sympathise with the efforts to let evolution as free as possible to find his own solution, we also agree with the maxim ascribed to Albert Einstein that "everything should be made as simple as possible, but no simpler". In our case, the fact that rewarding communication leads to effective behaviour but avoiding to do so leads just to failure widely justifies the use of such a procedure. Of course, the impact of such a choice depends on one's goals. If one's goals are technological, i.e. to develop teams of autonomous robots that are able to allocate different roles and behave accordingly for practical purposes, there is no reason to avoid the use any kind of expedient, let alone one that proves to be very effective. If one's goal is scientific, that is if one is interested in understanding role differentiation in real organism, the use of fitness components that directly reward communicative behaviours is more problematic, as it is not clear how such communicative behaviours per se may lead to higher chances of survival and reproduction. However, if one is not interested in understanding the evolution of role differentiation but rather how role differentiation can be realised in groups of homogeneous individuals, the way in which we get to our role-differentiating robots is not relevant and ours is just an effective way to obtain the behaviour that we want to investigate which otherwise could not be studied (because without rewarding communication role differentiation could not evolve). The detailed analysis of how our robots manage to solve the task is the first line of future research.

A second possible line of future research consists in trying to evolve other collaborative behaviours which have different kinds and numbers of roles. Our approach is not strictly related to the development of a group in which there is one individual that plays one role (the 'leader') and all the others play another role (the 'followers'). In fact, in previous work we have already shown that other ways of allocating roles are possible: for example, we were able to evolve groups which had to split in two evenly distributed subgroups, with half of the robots sending high-value signals and the other half sending low value signals. It will be interesting to demonstrate that other interesting non-communicative collaborative behaviours can be developed that require this or other kinds of role allocation.

Finally, another possible development consists in trying to release the designer from the need to specify the number and distribution of roles within the group by using information theoretic measures like Shannon entropy (Shannon, 1948) to reward robots for assuming different roles in different contexts while leaving the robots free to autonomously determine the right number and distribution of roles depending on the task at hand.

Acknowledgements

References

- Anderson, C. and Frank, N. (2001). Teams in animal societies. *Behavioral Ecology*, 12:534–540.
- Baldassarre, G., Nolfi, S., and Parisi, D. (2003). Evolving mobile robots able to display collective behavior. *Artificial Life*, 9:255–267.
- Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2013). Swarm robotics: a review from the swarm engineering perspective. pages 1–41.
- Dorigo, M., Trianni, V., Sahin, E., Gross, R., Labella, T., Baldassarre, G., Nolfi, S., Deneubourg, J.-L., Mondada, F., Floreano, D., and Gambardella, L. (2004). Evolving selforganizing behaviors for a swarm-bot. *Autonomous Robots*, 17(2-3):223–245.
- Floreano, D., Mitri, S., Magnenat, S., and Keller, L. (2007). Evolutionary conditions for the emergence of communication in robots. *Current Biology*, 17:514–519.
- Garnier, S., Gautrais, J., and Theraulaz, G. (2007). The biological principles of swarm intelligence. Swarm Intelligence, 1:3–31.
- Gigliotta, O., Mirolli, M., and Nolfi, S. (2009). Who is the leader? dynamic role allocation through communication in a population of homogeneous robots. In Serra, R., Villani, M., and Poli, I., editors, Artificial Life and Evolutionary Computation

- (Proceedings of Wivace 2008), pages 167–177. World Scientific Publishing.
- Gigliotta, O., Mirolli, M., and Nolfi, S. (Subm). Communication based dynamic role allocation in a group of homogeneous robots. *Natural Computing*.
- Harvey, I., Di Paolo, Ezequiel andTuci, E., Quinn, M., and Wood, R. (2005). Evolutionary robotics: A new scientific tool for studying cognition. *Artificial Life*, 11(1–2):79–98.
- Maynard-Smith, J. (1997). *The theory of evolution*. Cambridge University Press, Cambridge.
- Mirolli, M. and Nolfi, S. (2010). Evolving communication in embodied agents: Theory, methods, and evaluation. In Nolfi, S. and Mirolli, M., editors, *Evolution of Communication and Language in Embodied and Situated Agents*, pages 105–121. Springer, Berlin.
- Mirolli, M. and Parisi, D. (2005). How can we explain the emergence of a language which benefits the hearer but not the speaker? *Connection Science*, 17(3-4):325–341.
- Mirolli, M. and Parisi, D. (2008). How producer biases can favour the evolution of communication: An analysis of evolutionary dynamics. *Adaptive Behavior*.
- Mirolli, M. and Parisi, D. (2011). Towards a vygotskyan cognitive robotics: The role of language as a cognitive tool. *New Ideas in Psychology*, 29:298–311.
- Nolfi, S. (1998). Evolutionary robotics: Exploiting the full power of self-organization. *Connection Science*, 10(3-4):167–183.
- Nolfi, S. (2005). Emergence of communication in embodied agents: Co-adapting communicative and non-communicative behaviours. *Connection Science*, 17(3-4):231–248.
- Nolfi, S. and Floreano, D. (2000). *Evolutionary robotics. The biology, intelligence, and technology of self-organizing machines*. MIT Press, Cambridge, MA.
- Quinn, M., Smith, L., Mayley, G., and Husbands, P. (2003). Evolving controllers for a homogeneous system of physical robots: Structured cooperation with minimal sensors. *Philo-sophical Transactions of the Royal Society of London, Se*ries A: Mathematical, Physical and Engineering Sciences, 361:2321–2344.
- Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423.
- Spector, L., Klein, J., Perry, C., and Feinstein, M. (2005). Emergence of collective behavior in evolving populations of flying agents. In *Genetic Programming and Evolvable Machines*, 6, pages 111–125.
- Tuci, E., Mitavskiy, B., and Francesca, G. (2013). On the evolution of self-organised role-allocation and role-switching behaviour in swarm robotics: a case study. In *Advances in Artificial Life (Proceedings of ECAL2013)*, pages 379–386.

Evolving Autonomous Agent Controllers as Analytical Mathematical Models

Paul Grouchy¹ and Gabriele M.T. D'Eleuterio¹

¹University of Toronto Institute for Aerospace Studies, Toronto, Ontario, Canada M3H 5T6 paul.grouchy@mail.utoronto.ca

Abstract

A novel Artificial Life paradigm is proposed where autonomous agents are controlled via genetically-encoded Evolvable Mathematical Models (EMMs). Agent/environment inputs are mapped to agent outputs via equation trees which are evolved using Genetic Programming. Equations use only the four basic mathematical operators: addition, subtraction, multiplication and division. Experiments on the discrete Double-T Maze with Homing problem are performed; the source code has been made available. Results demonstrate that autonomous controllers with learning capabilities can be evolved as analytical mathematical models of behavior, and that neuroplasticity and neuromodulation can emerge within this paradigm without having these special functionalities specified a priori.

Introduction

When looking to design an Artificial Intelligence (AI) capable of robust and adaptable behavior, one must first choose how to represent such an agent *in silico*. Although many such representations exist, two of the most common are direct computer code representations, which can be programmed autonomously via Genetic Programming (GP) (Koza, 1992; Poli et al., 2008), and Artificial Neural Networks (ANNs), which are computational models based on the biological neural networks of animal brains that can be designed using one of a variety of approaches (Floreano et al., 2008).

Both of these representations require significant *a priori* design decisions. For a GP approach, one must choose which programming operations to include (e.g., bit-shift, and, or, xor, if-then-else, etc.), with a trade-off between potential program capabilities and search space size (and thus the ability of GP to find a solution). With ANNs, one must choose from many different types (e.g., Feedforward, Recurrent, Continuous-Time, Spiking, etc.), with trade-offs that have not yet been systematically investigated (Floreano et al., 2008). Other important design decisions for ANNs include the type of activation function and the incorporation of learning capabilities. Thus, to arrive at an agent representation appropriate for their task, a designer must first invest

a significant amount of time and computational resources. Furthermore, any such design decisions introduce additional experimenter bias into the simulation. Finally, complex behaviors require complex representations, adding to the already high computational costs of ALife algorithms while further obfuscating the inner workings of the evolved agents.

We propose Evolvable Mathematical Models (EMMs), a novel paradigm whereby autonomous agents are evolved via GP as analytical mathematical models of behavior. Agent inputs are mapped to outputs via a system of genetically encoded equations. Only the four basic mathematical operations (addition, subtraction, multiplication and division) are required, as they can be used to approximate any analytic function, thus eliminating the need to determine which operators to employ. More sophisticated operations can of course be added at the will of the designer; however, theoretically the four basic ones constitute the minimal set of operations required and accordingly the designer can be relieved of much of the guessing game. Furthermore, since agents are represented directly as mathematical equations, the evolved agents are amenable to mathematical analysis post facto.

The rest of this paper is organized as follows. In the next section, previous work on evolving ANNs with learning behaviors and on Genetic Programming for agent control is presented. This is followed by the algorithmic details of EMMs. A Double-T Maze problem that requires learning capabilities is then described, with the results of EMM experiments in this domain presented in the subsequent section. The source code for these experiments is provided. Finally, conclusions and future work will be discussed.

Background

Neuroplasticity and Neuromodulation

Autonomous agents are often evolved as artificial neural networks (ANNs). Typically, when one is evaluating an ANN, either in simulation or hardware, the ANN's connection weights are fixed. It is only when generating offspring genomes that genetic operators such as mutation can modify the ANN's weights. This is contrary to biological neural

networks, however, where "neuroplasticity" allows for connections between neurons to change during the lifespan of an organism (Pascual-Leone et al., 2005). Neuroplasticity (or "plasticity" for short) is what enables biological organisms to learn, modifying the way they react to certain inputs from the environment.

Hebbian learning, which is based on how learning is thought to occur in biology (Hebb, 1949), can be used to implement plasticity in ANNs (Floreano and Mondada, 1996). In this paradigm, local learning rules Δw_{ij} are evolved for each network connection weight w_{ij} . After each timestep t of an ANN's lifespan, each of its connection weights is updated using its associated learning rule

$$w_{ij}^{t+\Delta t} = w_{ij}^t + \eta \Delta w_{ij}^{t+\Delta t} , \qquad (1)$$

where $0 \le \eta \le 1$ is the evolvable learning rate. An example learning rule Δw_{ij} is the "plain Hebb rule," defined as

$$\Delta w_{ij}^{t+\Delta t} = (1 - w_{ij}^t) v_i^t v_j^t , \qquad (2)$$

where v_i^t is the output of neuron i at the current timestep.

In the learning paradigm described above, connection weights are adjusted at every timestep throughout the lifetime of an agent. This differs from biological systems which are theorized to use "neuromodulation" to control and stabilize learning (Bailey et al., 2000).

To improve learning algorithm performance, the Analog Genetic Encoding (AGE) algorithm, an encoding method based on biological gene regulatory networks that can evolve both the connections weights and structure of an ANN (Mattiussi and Floreano, 2004; Dürr et al., 2006), was modified to allow for Hebbian learning which could be enabled and disabled via neuromodulatory signals (Soltoggio et al., 2007; Durr et al., 2008; Soltoggio et al., 2008). In this approach, a generalized Hebbian rule from (Niv et al., 2002) was modified to include a modulatory signal m:

$$\Delta w_{ij}^{t+\Delta t} = m^t \eta \left(A v_i^t v_j^t + B v_i^t + C v_j^t + D \right)$$
 (3)

where A, B, C and D are evolvable parameters that determine the importance of the different types of Hebbian learning and $0 \le m^t \le 1$ is the current strength of the signal produced by one or more special modulatory neurons. These modulatory neurons operate in a manner similar to regular neurons, taking inputs from other neurons in the network through network connections and generating their output value m using an activation function.

Fixed-Weight Learning

Learning behaviors have been observed in several fixed-weight ANN-based experiments as well. Fixed-weight recurrent neural networks seem to be able to accomplish certain tasks requiring learning, as the recurrent neural connections can act as a type of memory (e.g., (Stanley et al., 2003;

Soltoggio et al., 2008)). Continuous-time recurrent neural networks (CTRNNs) have also demonstrated learning capabilities, sometimes even outperforming plastic neural networks (Jesper and Floreano, 2002; Tuci and Quinn, 2003).

Genetic Programming for Agent Control

Genetic Programming (GP) can be used to evolve autonomous controllers as computer programs. In the first such experiment (Koza and Rice, 1992), controllers were represented as variable-length trees containing sensor inputs and four preprogrammed macros, such as "if-less-than-or-equal." Programs that could control a simulated robot to find a box in an irregularly shaped world and push it to a wall from four different starting configurations were evolved.

A linear implementation of GP was used to evolve machine code to control a Khepera robot in (Nordin and Banzhaf, 1995, 1997). This work was recently extended in (Burbidge et al., 2009; Burbidge and Wilson, 2014), where machine code was evolved using Grammatical Evolution. Here, agent genomes are binary strings that are mapped to their machine code phenotypes via a prespecified generative grammar.

GP has also been used to evolve competitors for the RoboCup robotic soccer tournament. The team "Darwin United" was evolved using GP with a variety of possible operations, including basic mathematical operators, reading and writing to memory locations and executing a variety of programmer-designed subroutines (Andre and Teller, 1999). A simple GP approach was also used to evolve robot goalie behaviors in (Adorni et al., 1999). This work is the most similar to our Evolvable Mathematical Models paradigm presented below, as it is evolving mathematical equations as trees for robot control. However, these experiments were performed on a relatively simple task (especially considering the amount of information provided to the controller), had only one equation tree/agent output, employed two additional operators (sine and cosine), and did not allow for extra state variables/equation trees to be evolved.

Evolvable Mathematical Models

Our Evolvable Mathematical Models (EMM) algorithm uses Genetic Programming (GP) to evolve mathematical models of behavior. An earlier version of this algorithm that evolved one ODE per agent output (i.e., there were no extra state variables) was presented in (Grouchy and D'Eleuterio, 2010) and implemented in (Grouchy and Lipson, 2012). The core idea is that one can evolve autonomous agent controllers as mathematical equations that map from agent inputs to outputs.

Representation

An EMM-based agent is represented as a system of equations, with one equation for each of the N experimenter-defined outputs v_i in the simulation. Additional "extra"

equations can be added through an "add-equation" mutation (similar to the concepts of Automatically Defined Functions and Architecture-Altering Operations (Koza, 1994)). An agent's N' extra equations do not have associated agent outputs; however, they modify agent outputs indirectly via their incorporation into those equations that affect agent outputs directly. Therefore, an agent is fully specified by its system of state equations

$$\mathbf{v}^{t+\Delta t} = \mathbf{f} \left(\mathbf{u}^t, \mathbf{v}^t \right) \tag{4}$$

and its evolvable initial conditions $\mathbf{v}^{t=0}$. Here,

$$\mathbf{v} = [v_1, v_2, ..., v_N, v_{N+1}, ..., v_{N+N'}]^{\mathrm{T}}$$
 (5)

and

$$\mathbf{u} = [u_1, u_2, ..., u_M]^{\mathrm{T}} , \qquad (6)$$

where M is the number of experimenter defined inputs to the agent and

$$\mathbf{f}\left(\mathbf{u},\mathbf{v}\right) = \left[f_1\left(\mathbf{u},\mathbf{v}\right), f_2\left(\mathbf{u},\mathbf{v}\right), ..., f_{N+N'}\left(\mathbf{u},\mathbf{v}\right)\right]^{\mathrm{T}}$$
 (7)

are the agent's genetically encoded state functions.

For all experiments presented here, tree structures were used to represent the EMMs in the agent genome, as in canonical GP. Only the four basic mathematical operators are allowed (addition, subtraction, multiplication and division), from which any analytic function can be approximated. A real-valued variable-length vector was used to represent an agent's initial conditions $\mathbf{v}^{t=0}$ in the genome.

A genome contains N+N' equation trees, one for each output and extra state variable of the system. Each tree contains a collection of terminal and nonterminal nodes. The set of possible terminal nodes is comprised of all possible real constants and variables (i.e., input, output and extra state), while the set of possible nonterminal nodes is composed of addition, subtraction, multiplication and division. The number of "child" nodes (subtrees) of a nonterminal node is two, as all four of the basic operations have an arity of two. An example genome for an agent with N+N'=2 is shown in Figure 1.

Evolution

Initialization. The random initialization of the N initial equation trees in each initial genome (e.g., at generation 0 in a genetic algorithm) is done using the "ramped half-and-half" method from GP. This method is a combination of two methods, the "full" and "grow" methods. For both methods, a maximum depth (i.e., the maximum number of edges that need to be traversed to reach a node, starting from the root node) is specified. In the "full" method, nonterminal (i.e., operator) nodes are randomly generated until the maximum depth is reached. At the maximum depth, only terminal (i.e., operand) nodes are created. In the "grow" method,

$$v_1^{t+\Delta t} = f_1\left(\mathbf{u}^t, \mathbf{v}^t\right) = \underbrace{1}_{v_1^t} + \underbrace{v_1^t}_{v_1^t} = 1 + v_1^t$$

$$v_2^{t+\Delta t} = f_2\left(\mathbf{u}^t, \mathbf{v}^t\right) = \underbrace{v_1^t}_{v_1^t} + \underbrace{v_1^t}_{v_1^t}$$

$$= v_1^t / (2 + u_1^t)$$

Figure 1: An example EMM for an agent with $N+N^\prime=2$. The two trees, along with the two initial values $v_1^{t=0}$ and $v_2^{t=0}$ (not shown), are how the agent's EMM is encoded in its genome.

as in the "full" method, only terminal nodes are created at the maximum depth. The difference is that before the maximum depth is reached, randomly generated nodes can be either terminal or nonterminal nodes (with equal probability), allowing for a wider range of potential tree shapes. For all experiments presented here, half of the trees are generated with a maximum depth of 1, while the other half have a maximum depth of 2. Terminal nodes are set to a randomly chosen variable or a random constant with equal probability. Initial genomes do not contain any extra state variables, i.e., N'=0. Initially, constants are randomly selected and initial output values $\mathbf{v}^{t=0}$ are set to random values.

Sexual Recombination. Sexual recombination allows for large jumps in the search space through combining two partial solutions. Any such genetic operation requires two parents to produce an offspring genome. Otherwise, a single parent's genome is cloned to produce an offspring genome. In EMMs, tree-level sexual recombination occurs in a fashion similar to crossover in GP.

An offspring genome is initially generated as a clone of the first parent. If a tree in the offspring genome is selected to undergo tree-level recombination, one of its nodes is selected at random and replaced with a randomly selected subtree from the second parent. Randomly selected nodes have a probability of 0.1 of being terminal nodes. If there are subtrees below the selected node in the offspring's tree, they are discarded. The subtree from the second parent can come from any of its equation trees. This allows for partial solutions to be reused and to be copied to different equation trees. This subtree grafting operation is similar to the subtree mutation operation described below. If the subtree being copied over references extra state variables that are not present in the offspring genome, the equations for those state variables are copied to the offspring genome from the second parent. An offspring genome is subject to a variety of genetic mutations, even if it has been produced via sexual recombination.

Tree Mutation. Tree mutations (as well as extra state variable mutations and sexual recombination) can occur when a parent genome is being copied to its offspring. This means that mutations can only occur *between* generations. Agent genomes remain fixed throughout an agent's lifetime.

These mutations are implemented in a manner similar to GP. If a tree is selected to be mutated, one of a variety of mutations is applied:

- **Point Mutation.** A point mutation performs one of several operations on a single randomly selected node in the tree:
 - Perturbation of a constant. This operation can only be performed if the tree in question contains one or more constants. The operation adds a random value to a randomly selected constant.
 - Mutation of a nonterminal node. This operation is performed if a perturbation of a constant was not done and if a randomly selected node is a nonterminal. A new nonterminal operation is randomly selected from the set of addition, subtraction, multiplication and division.
 - Mutation of a terminal node. This operation is performed if a perturbation of a constant was not done and if a randomly selected node is a terminal. One of two mutations occurs, with equal probability. The chosen terminal is either mutated to a randomly chosen variable or it is mutated to a random constant.
- Subtree Mutation. This operation selects a random node on the original tree and replaces it with a new random subtree. This new subtree is generated in an identical fashion to initial trees (see "Initialization" above). A small variation to the standard subtree mutation was also added. With a given probability, the roles of the subtree and the original tree are swapped, i.e., a random node on the randomly generated subtree is replaced with the entire original tree and this becomes the new tree of the offspring.

Add-Equation Mutation. An offspring is subject to "addequation" mutations, as well as those mutations described above. This mutation produces a new equation tree of depth 1 or 2 using the "ramped half-and-half" method described previously. A new variable v_j , j > N is added to \mathbf{v} . This is the variable that the new equation tree will be modifying. The new variable v_j is also randomly incorporated into a randomly selected existing equation, either through a point mutation or a subtree mutation (with equal probability), as described above. Finally, $v_j^{t=0}$ is set to a random value.

Initial-Value Mutation. An offspring's initial values $\mathbf{v}^{t=0}$ are subject to mutation as well. If an initial value is to be mutated, it will either be set to a new random value or perturbed

by a value drawn from a given distribution. These two types of initial value mutations occur with equal probability.

Equation Reduction. When an offspring genome is produced, it is checked for possible equation simplifications with a probability of 0.1. For example, the computation 0+1 will be reduced to 1. Results in (Grouchy and D'Eleuterio, 2010) demonstrated that this improves both solution quality and execution times.

If an extra state variable is not referenced anywhere in the offspring genome (excluding the variable's own equation tree), that variable and its corresponding equation tree are discarded.

Execution

An EMM-based agent's behavior over the course of its lifetime is determined as follows:

- 1. Set t = 0
- 2. Set $v_i = v_i^{t=0}$, i = 1, ..., N, ..., N + N'
- 3. Update \mathbf{u}^t with current agent inputs (i.e., current sensor values)
- 4. Evaluate $\mathbf{f}(\mathbf{u}^t, \mathbf{v}^t)$
- 5. Update agent output and extra state variables $\mathbf{v}^{t+\Delta t} = \mathbf{f}(\mathbf{u}^t, \mathbf{v}^t)$
- 6. Run agent for Δt timesteps using agent output values $v_i^{t+\Delta t},\,i=1,...,N$
- 7. Set $t = t + \Delta t$
- 8. Go to step 3 (unless the end of the agent's lifespan has been reached)

The above steps apply to agents operating in a simulation environment as well as to embodied robotic agents operating in the real world. Steps 3-6 are equivalent to propagating agent inputs through an ANN to produce agent outputs for a single timestep of an agent's lifetime in an ANN-based experiment.

Double-T Maze Experiments

In its simplest form, a T Maze test consists of a series of trials where a robot starts in the home position, choses one of the two "arms" of the maze to visit and collects the reward at the end of that arm. In some cases, the robot is automatically returned home once the end of the maze is reached, whereas in others part of the task is for the agents to find their own way home. For each trial, one arm of the maze contains a high reward, while the other contains a low one. The purpose of this task is to demonstrate learning. A successful agent should search both arms for the high reward, and then return to the high-reward arm of the maze in subsequent trials. If the reward is moved, the successful agent should search for and relearn its new position. A Double-T Maze has four arms instead of two, while still only having one high reward (Figure 2). In a discrete maze environment,

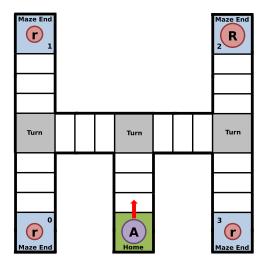


Figure 2: A discrete Double-T Maze. "A" is the agent, "r" are low rewards and "R" is the high reward.

agents must decided to move straight for one unit, or turn left or right, requiring only a single output variable.

The specific T Maze chosen for these experiments was the discrete Double-T Maze with Homing requirements used in (Soltoggio et al., 2008), although certain experimental settings may vary slightly owing to a lack of details in the original paper and a lack of available source code. This version was selected for several reasons. This problem domain has only been solved with plastic ANNs; fixed-weight ANNs have so far been unsuccessful. Furthermore, the Double-T Maze is very difficult, requiring agents to choose repeatedly one of four different movement patterns depending on where they find the high reward. If agents have yet to discover the location of the high reward, they must search up to four different maze arms (requiring four different movement patterns) sequentially. Adding to the difficulty is the requirement that agents must return home after reaching an end of the maze. This doubles the size of each of the four movement patterns. For example, to get to the top left part of the maze an agent must turn left, then right. To then return home, the agent must turn left, then right again. There is a unique four-turn pattern for each of the four arms (LL-RR, LR-LR, RL-RL, RR-LL).

The main point of this experiment is to demonstrate that EMMs can solve this challenging task. However, we attempt to tackle this problem using the same number of fitness evaluations as in (Soltoggio et al., 2008).

Problem Definition. Agent fitness is evaluated over a series of trials. For each trial, the agent is evaluated for a maximum of 35 steps, with each step consisting of one evaluation of the agent's equations and the execution of one move (forward or turn) based on the agent's output v_1 . A trial begins

with the agent at the "home" position. If an agent executes a turn command while not on a "turn" position (i.e., while on the home position, one of the four reward positions or in a corridor) or executes three consecutive "move forward" commands on a turn position, this is considered a "crash." Crashes end the current trial, returning agents to the home position and subtracting 0.4 from their total fitness. If an agent completes a trial without returning to the home position, a penalty of 0.3 is applied to their total fitness. If the agent reaches one of the three low-reward arms of the maze, a score of 0.2 is added to their fitness. The high reward arm yields a fitness boost of 1.0. When an agent reaches the end of a maze arm, it is automatically turned 180°. Corridors and turn points last for three forward steps each.

Agents have access to four inputs, "turn," "maze end," "home" and "reward" ($u_1,\,u_2,\,u_3$ and u_4 , respectively). The turn input is set to 1.0 when the agent is on a turning point, 0.0 otherwise. The maze-end input is set to 1.0 when the agent is at the end of one of the four maze arms, 0.0 otherwise. The home input is set to 1.0 when the agent is at the home position, 0.0 otherwise. Finally, if the agent collects a low reward, the reward input is set to 0.2 for one step. Collecting a high reward sets the reward input to 1.0 for one step. The reward input is 0.0 at all other times.

Agents have one output v_1 . If $v_1 < -0.33$, the agent performs a left turn and then moves forward one unit. If $v_1 > 0.33$, the agent performs a right turn and then moves forward one unit. Otherwise, the agent moves forward one unit in its current direction. All inputs and references to variables v_i , i = 1, ..., N + N' are subject to noise by adding a random value taken from the uniform distribution [-0.005, 0.005] at each equation evaluation.

Agent fitness is evaluated on a set of 200 trials, with the high reward randomly positioned for the first trial. The high reward is randomly repositioned after a randomly selected number of trials H_t , with $35 \leq H_t \leq 65$. Reward repositioning happens three to four times per 200 trial run, with H_t being regenerated after every repositioning. During the evolutionary runs, the first four high reward positions were forced to be distinct. This was not enforced for the 100 sets of 200 trials used for testing top agents.

EMM Algorithm Details. For this task, an island model is used¹. Each of the 10 islands has a population of 100, giving a total population size of 1,000. Islands are arranged in a ring formation, with the top agent from each island being copied (migrating) to the left or right island every 20 generations. The direction of migration is constant across all islands and switches after each migration. Each island population is tested on its own set of 200 trials, with all 10 sets being regenerated after each generation. Tournament selec-

¹The source code for these experiments is available for download at http://www.sr.utias.utoronto.ca/images/downloads/grouchy_alife_2k14.zip.

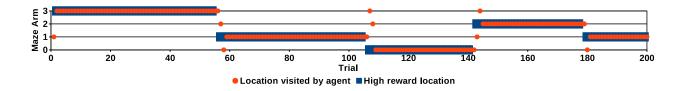


Figure 3: High reward location and maze end visited by a successful EMM agent for each trial of a 200 trial run. Locations 0,1,2,3 are indicated in Figure 2.

tion with a tournament size of 15 is used, and each island's top agent is cloned for the next generation (elitism).

When random constants are needed, they are selected from the uniform distribution [-5, 5]. Random initial values $\mathbf{v}^{t=0}$ are set to values from the uniform distribution [-1, 1].

Offspring genomes are produced as a clone of a single parent with a probability of 0.3, otherwise tree-level sexual recombination can occur on one or more of the offspring's N+N' trees with a probability of $0.5/\left(N+N'\right)$ per tree. Tree mutations can happen to any offspring genome with a probability of 0.1 per offspring tree, whereas an extra state variable/equation tree is also added to a genome with a probability of 0.1 per previously existing offspring tree. Offspring are required to undergo at least one tree or addequation mutation. An initial value is mutated with a probability of $0.1/\left(N+N'\right)$.

If a tree mutation is to occur, it will be a point mutation with a probability of 0.5, otherwise it will be a subtree mutation. If a point mutation is to occur and the tree in question contains at least one constant, a perturbation of a constant mutation will happen with a probability of 0.5, adding a random value taken from a Gaussian distribution with mean 0 and standard deviation 0.5 to a randomly selected constant. During subtree mutation, the original tree and the randomly generated subtree are swapped with a probability of 0.05. Initial values are perturbed using a value selected from a Gaussian distribution with mean 0 and standard deviation 0.25.

Output values v_1 are capped to the range [-1,1], however extra state variables v_j , j=2,...,N' are unbounded. If there is a zero-divided-by-zero operation, or if any variable exceeds the minimum or maximum allowable values of the programming language being used, the current trial is terminated. A maximum genome size of 200 nodes was imposed across all experiments.

At the end of every generation, the top agent from each island is tested on a fixed test set of 100 randomly generated 200 trial runs. Each experiment is run for 1,000 generations, and the final result is taken to be the agent that performs the best on the test set. This is notably different than the experiments in (Soltoggio et al., 2008) where only the top agent in the final generation is tested.

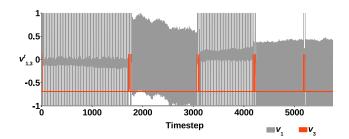


Figure 4: v_1^t and v_3^t values over time for the same EMM agent and 200 trial run as shown in Figure 3.

Algorithm	Test Score		# Successful
	μ	σ	
10 islands w/o extra eqns	95	10	0
10 islands w/ extra eqns	172	18	9
40 islands w/ extra eqns	186	5	18

Table 1: Double-T Maze results from 50 evolutionary runs. A "successful run" has occurred if an agent scores 189.4 or higher on the test set. μ is the mean and σ is the standard deviation.

Results

For the test set used in these experiments, we calculated the "worst-case perfect test score" to be 189.4 (the theoretical maximum test score was calculated to be 197.24). This value is the average score of a perfect agent across the 100 test runs, assuming the agent always searches each low-reward arm once before finding the high reward (hence "worstcase") and always returns to the high-reward arm once it is discovered (hence "perfect agent"). Thus we consider an agent with a test score of 189.4 or higher to be a solution to this Double-T Maze. Table 1 shows the results from 50 runs using the same test set, but with different initial populations and different training sets. The runs with 10 islands use the same number of fitness evaluations as in (Soltoggio et al., 2008), although with significantly more agents tested. Results with 10 islands and extra equations/state variables disabled are shown, demonstrating significantly worse performance and an inability to fully solve the task. Experiments with 40 islands are also reported, demonstrating performance improvements given more fitness and test evaluations.

From these results, we can conclude that EMMs can successfully solve this difficult task requiring learning. A successful agent with a test score of 191.884 will be examined further. Figure 3 shows how this agent performs on a single 200-trial run. The location of the high reward is shown for each trial, as well as the maze arm visited by the agent. The agent's foraging pattern seems to be fixed: arm 1, arm 3, arm 2, arm 0, then back to arm 1 and the pattern repeats.

The full system of equations (with rounding and simplifications) of this top agent is

$$\begin{array}{rcl} v_1^{t+\Delta t} & = & u_1^t v_2^t v_4^t \\ v_2^{t+\Delta t} & = & 5.92 v_3^t / v_2^t \end{array} \tag{8}$$

$$v_2^{t+\Delta t} = 5.92v_3^t/v_2^t (9)$$

$$\begin{array}{lll} v_2^{t+\Delta t} &=& 5.92v_3^*/v_2^* & (9) \\ v_3^{t+\Delta t} &=& u_2^t - u_4^t - 0.69 & (10) \\ v_4^{t+\Delta t} &=& -0.41u_1^t v_7^t & (11) \\ v_5^{t+\Delta t} &=& -0.33v_6^t & (12) \\ v_6^{t+\Delta t} &=& 0.65 - v_8^t - (1.18/v_3^t) & (13) \end{array}$$

$$v_4^{t+\Delta t} = -0.41 u_1^t v_7^t \tag{11}$$

$$v_5^{t+\Delta t} = -0.33v_6^t \tag{12}$$

$$v_6^{t+\Delta t} = 0.65 - v_8^t - (1.18/v_3^t)$$
 (13)

$$v_7^{t+\Delta t} = v_5^t - 0.48 (14)$$

$$v_8^{t+\Delta t} = v_9^t (0.02v_9^t + 0.05v_{10}^t - 0.23)$$
 (15)

$$-v_5^t - 0.14v_{10}^t + 0.48$$

$$v_9^{t+\Delta t} = 56.11u_2^t (16)$$

$$v_9^{t+\Delta t} = 56.11u_2^t$$
 (16)
 $v_{10}^{t+\Delta t} = 0.80 - v_4^t$ (17)

Note that v_1 is the agent's output variable and evolved initial conditions $\mathbf{v}^{t=0}$ are omitted. Figure 5 shows the relationships between variables within this agent's evolved equations. The only equation containing the variable u_4 (the "reward" input) is (10), and its corresponding state variable v_3 seems to be modulating learning. The value of v_3^t is constant at -0.69, except in the cases where an agent is at a maze end $(u_2^t=1)$ and a low reward is collected $(u_4^t=0.2)$. In these cases, $v_3^{t+\Delta t} = 1 - 0.2 - 0.69 = 0.11$. Figure 4 shows the agent's behaviors (i.e., its output values v_1) and the values of v_3 over all timesteps from the same 200 trial run shown in Figure 3. One can clearly see the neuromodulation-like behavior of v_3 , as the agent output patterns (v_1) do not change when $v_3^t = -0.69$, however these patterns do change when $v_3^t = 0.11$. A positive spike of v_3 seems to cause the agent to try the next arm in its forage pattern (learning), while a fixed negative v_3 value causes the agent to revisit the same arm (no learning). Thus neuromodulation-like behavior has evolved without special neural structures having been specified a priori.

Conclusions

We have presented a novel Artificial Life paradigm that uses Evolvable Mathematical Models (EMMs) as controllers for autonomous agents. A Genetic Programming algorithm

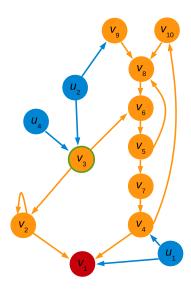


Figure 5: Relationship between variables within the successful EMM agent's evolved equations. Inputs are shown in blue, the agent's output v_1 is red and the extra state variables are orange. Data require one timestep to traverse an orange arrow, whereas input data traverse blue arrows instantaneously. The state variable v_3 is emphasized in green as it plays the role of neuromodulator. The calculations performed at each node are shown in (8) to (17).

was used to evolve systems of equations that map agent inputs to outputs. Functions are represented in the genome as variable-length trees, one for each agent output, and are composed of the four basic mathematical operators, addition, subtraction, multiplication and division, as well as input and output variables and constants. These EMMs can approximate any analytic function. Further trees and corresponding extra state variables can be added through an "add-equation" mutation. Experiments were performed on the challenging Double-T Maze with Homing domain, a task previously only solved using artificial neural networks with connection weight plasticity. Solutions to this domain were evolved successfully using fixed-structure EMMs and the same number of fitness function evaluations as a previous ANN-based experiment. These solutions demonstrated neuromodulation-like learning behaviors without any special neuroplasticity or neuromodulation structures having been specified a priori. Furthermore, evolved solutions are readily examinable, as they are represented directly as mathematical equations and are thus amenable to mathematical analysis.

The work presented here is intended as a first step towards evolving autonomous agents as mathematical models of behavior. Future work should look to improve the evolvability of EMMs through alternative genetic encodings (e.g., Linear GP) and different types of evolutionary search. The results presented here demonstrate the ability of EMMs to evolve behaviors similar to those produced by neuromodulated plastic neural networks. A question that then arises is whether EMMs can be evolved to model behaviors produced by other types of neural networks, such as biological neural networks. Further experimentation on even more challenging domains are required to explore the effectiveness of Evolvable Mathematical Models as autonomous agent controllers.

Acknowledgements

This work was supported by the Natural Sciences and Engineering Research Council of Canada.

References

- Adorni, G., Cagnoni, S., and Mordonini, M. (1999). Genetic programming of a goal-keeper control strategy for the robocup middle size competition. In *Genetic Programming*, pages 109–119. Springer.
- Andre, D. and Teller, A. (1999). Evolving team darwin united. In *RoboCup-98: Robot soccer world cup II*, pages 346–351. Springer.
- Bailey, C. H., Giustetto, M., Huang, Y.-Y., Hawkins, R. D., and Kandel, E. R. (2000). Is heterosynaptic modulation essential for stabilizing hebbian plasiticity and memory. *Nature Reviews Neuroscience*, 1(1):11–20.
- Burbidge, R., Walker, J. H., and Wilson, M. S. (2009). Grammatical evolution of a robot controller. In *Intelligent Robots and Systems*, 2009. IROS 2009. IEEE/RSJ International Conference on, pages 357–362. IEEE.
- Burbidge, R. and Wilson, M. S. (2014). Vector-valued function estimation by grammatical evolution for autonomous robot control. *Information Sciences*, 258:182–199.
- Dürr, P., Mattiussi, C., and Floreano, D. (2006). Neuroevolution with analog genetic encoding. In *Parallel Problem Solving from Nature-PPSN iX*, pages 671–680. Springer.
- Durr, P., Mattiussi, C., Soltoggio, A., and Floreano, D. (2008). Evolvability of Neuromodulated Learning for Robots. In *The* 2008 ECSIS Symposium on Learning and Adaptive Behavior in Robotic Systems, pages 41–46, Los Alamitos, CA. IEEE Computer Society.
- Floreano, D., Durr, P., and Mattiussi, C. (2008). Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1:47–62.
- Floreano, D. and Mondada, F. (1996). Evolution of Plastic Neurocontrollers for Situated Agents. In 4th International Conference on Simulation of Adaptive Behavior (SAB'1996). MA: MIT Press.
- Grouchy, P. and D'Eleuterio, G. M. T. (2010). Supplanting neural networks with ODEs in evolutionary robotics. In *Simulated Evolution and Learning*, pages 299–308. Springer.
- Grouchy, P. and Lipson, H. (2012). Evolution of self-replicating cube conglomerations in a simulated 3D environment. In *Artificial Life*, volume 13, pages 59–66.

- Hebb, D. O. (1949). The Organization of Behavior: A Neuropsychological Theory. Wiley, New York.
- Jesper, B. and Floreano, D. (2002). Levels of dynamics and adaptive behavior in evolutionary neural controllers. In From Animals to Animats 7: Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior, volume 7, page 272. MIT Press.
- Koza, J. R. (1992). Genetic programming: on the programming of computers by means of natural selection (complex adaptive systems).
- Koza, J. R. (1994). Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press, Cambridge Massachusetts.
- Koza, J. R. and Rice, J. P. (1992). Automatic programming of robots using genetic programming. In AAAI, volume 92, pages 194–207.
- Mattiussi, C. and Floreano, D. (2004). Evolution of analog networks using local string alignment on highly reorganizable genomes. In Evolvable Hardware, 2004. Proceedings. 2004 NASA/DoD Conference on, pages 30–37. IEEE.
- Niv, Y., Joel, D., Meilijson, I., and Ruppin, E. (2002). Evolution of reinforcement learning in uncertain environments: A simple explanation for complex foraging behaviors. *Adaptive Behavior*, 10(1):5–24.
- Nordin, P. and Banzhaf, W. (1995). Genetic programming controlling a miniature robot. In Working Notes for the AAAI Symposium on Genetic Programming, pages 61–67. Citeseer.
- Nordin, P. and Banzhaf, W. (1997). An on-line method to evolve behavior and to control a miniature robot in real time with genetic programming. *Adaptive Behavior*, 5(2):107–140.
- Pascual-Leone, A., Amedi, A., Fregni, F., and Merabet, L. B. (2005). The plastic human brain cortex. *Annu. Rev. Neurosci.*, 28:377–401.
- Poli, R., Langdon, W. B., and McPhee, N. F. (2008). *A Field Guide to Genetic Programming*. Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk. (With contributions by J. R. Koza).
- Soltoggio, A., Bullinaria, J. A., Mattiussi, C., Dürr, P., and Floreano, D. (2008). Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. In AL-IFE, pages 569–576.
- Soltoggio, A., Durr, P., Mattiussi, C., and Floreano, D. (2007). Evolving Neuromodulatory Topologies for Reinforcement Learning-like Problems. In *IEEE Congress on Evolutionary Computation (CEC 2007) 25-28 Sept. 2007*, pages 2471–2478. IEEE Press.
- Stanley, K. O., Bryant, B. D., and Miikkulainen, R. (2003). Evolving adaptive neural networks with and without adaptive synapses. In *Evolutionary Computation*, 2003. CEC'03. The 2003 Congress on, volume 4, pages 2557–2564. IEEE.
- Tuci, E. and Quinn, M. (2003). Behavioural plasticity in autonomous agents: a comparison between two types of controller. In *Applications of Evolutionary Computing*, pages 661–672. Springer.

Attraction basins in a lac operon model under different update schedules

Marco Montalva¹, Gonzalo A. Ruz¹ and Eric Goles¹

¹Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibáñez, Av. Diagonal Las Torres 2640, Santiago, Chile marco.montalya@uai.cl

Abstract

In (Veliz-Cuba and Stigler, 2011) the authors proposed a Boolean model for the lac operon in Escherichia coli that is capable of predicting the operon being ON, OFF and bistable when the update schedule is the parallel one. We complement this work by using theoretical and algorithmic tools that allow us to know which are the configurations that converge to a fixed point or limit cycle (set namely attractor basin) for each deterministic update schedule. We show that, when bistability appears, about 70% of the dynamics have only the steady states ON and OFF. This latest having an attractor basin of an average size about 8 times bigger than that of ON. In the other 30%, the proportion is balanced between ON/OFF basins but the basins of limit cycles sum up, in average, about 5 times more than that of ON and OFF respectively. The techniques presented in this work are general and can be used to analyze other Boolean models.

Introduction

The *lac* operon in *Escherichia coli* is one of the earliest examples of an inducible system of genes being under both positive and negative control. This system is responsible for the metabolism of lactose in the absence of glucose and is known to exhibit bistability, in the sense that the operon is either induced (ON) or uninduced (OFF). In (Veliz-Cuba and Stigler, 2011) the authors proposed a Boolean model for it, showed in Fig. 1, where G_e and (L_e, L_{em}) are parameters (representing different concentration levels of extracellular glucose and lactose respectively) so that when certain values are assigned (L_e and L_{em} assigned stochastically) and considering that the order in which the nodes are updated (concept namely update schedule) is always the parallel one, the model is capable to predict the operon being ON, OFF and bistable depending of the fixed points (steady states) obtained, results that matches very well with the experimental data.

However, this model does not answer other dynamical question that can provide a more refined qualitative description of the *lac* operon; given a dynamic obtained with a specific update schedule, how are its attraction basins?

In this work, we address the previous question focusing on all the deterministic update schedules (an exponential

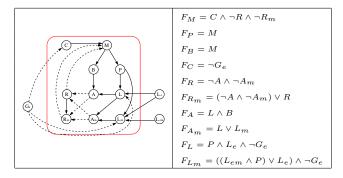


Figure 1: The *lac* operon network (left) and its local functions (right) proposed in (Veliz-Cuba and Stigler, 2011). Dashed/solid edges represent inhibitions/activations.

number). The idea is to make an exhaustive analysis of all these dynamics but not by a brute force process (i.e., list all them, each composed of 2^{10} configurations and then calculate each attraction basin) but through the algorithm of (Aracena et al., 2013) specially adapted for the *lac* operon network that efficiently enumerates the set of all update schedules s with the property of being representatives of an equivalence class whose elements (other update schedules s') have exactly the same dynamics that s. Thus, the number of dynamics to analyze decreases dramatically. These tools have shown to be effective in other models such as the Yeast and Mammalian cell cycle networks studied in (Goles et al., 2013) and (Ruz et al., 2014).

The lac operon model background

In model of Fig. 1, a configuration dynamics is represented the by $(M, P, B, C, R, R_m, A, A_m, L, L_m) \in \{0, 1\}^{10}.$ G_e parameter can be in two states; low $(G_e = 0)$ or high $(G_e = 1)$ while that L_e and L_{em} can be in three states; low, medium or high which is represented by $(L_e, L_{em}) = (0,0), (0,1)$ and (1,1) respectively. The operon is OFF when the value of the triple (M, P, B) is (0,0,0) and ON when (M,P,B) = (1,1,1). Under the

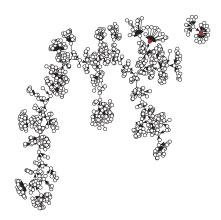


Figure 2: State transition graph of the *lac* operon model under the parallel update (bistability appears). The size of the attraction basins are 1006 and 18 for the steady states OFF and ON (in red), respectively.

parallel update schedule there are 4 possible cases:

Case 1: $G_e = 1$. All configurations eventually reach the unique steady state (0,0,0,0,1,1,0,0,0,0) (operon being OFF).

Case 2: $G_e = L_e = L_{em} = 0$. All configurations eventually reach the unique steady state (0,0,0,1,1,1,0,0,0,0) (operon being OFF).

Case 3: $G_e = L_e = 0 \wedge L_{em} = 1$. All configurations eventually reach one of the two steady states; (0,0,0,1,1,1,0,0,0,0) or (1,1,1,1,0,0,0,1,0,1) (operon being OFF and ON, respectively). That is, the model is bistable (see Fig. 2).

Case 4: $G_e = 0 \wedge L_e = L_{em} = 1$. All configurations eventually reach the unique steady state (1, 1, 1, 1, 0, 0, 1, 1, 1, 1) (operon being ON).

Analysis and discussion

For cases 1, 2 and 4 we proved the following Theorem:

Theorem 1. Cases 1, 2 and 4 are valid for every deterministic update schedule.

In case 3 we adapt the algorithm described in the introduction and the results are summarized in Table 1.

Theorem 1 guarantees that attraction basins of cases 1, 2 and 4 are the full state space (1024 configurations) whatever the update schedule. In Case 3 (bistability) about 70% of the dynamics have only the steady states ON and OFF. This latest having an attractor basin of average size about 8 times bigger than that of ON. In the other 30%, the proportion is balanced between ON/OFF basins but basins of limit cycles sum up, in average, about 5 times more than that of ON and OFF respectively. Fig. 3 is an example of a dynamic that has 3 limit cycles of length 4 plus one of length 2 (note that the update schedule is slightly different from the parallel one).

Case 3: $G_e = L_e = 0 \wedge L_{em} = 1$ (Bistability)					
Attractors	S	FP	LC		
OFF	684.7	908.9	153.7		
ON	124.4	115.1	146.5		
Limit cycles	214.9	0	723.8		

Table 1: Basin average size for case 3 calculated over S (the full set of deterministic updates schedules), $FP \subseteq S$ (those whose dynamic have only fixed points) and $LC \subseteq S$ (those whose dynamic have limit cycles), where |S| = 102,247,563 (100%), |FP| = 71,891,966 (70.3%) and |LC| = 30,355,597 (29.7%).



Figure 3: State transition graph of the *lac* operon model when the update schedule considered is such that $(M, P, B, C, R, A, L, L_m)$ are updated first simultaneously and then (R_m, A_m) simultaneously. The size of the attraction basins are 98 and 18 for the steady states OFF and ON (in red) respectively, and 908 for the limit cycles (in blue).

Acknowledgements

This work was supported by Fondecyt 3130466 (M.M.), Fondecyt 11110088 (G.A.R.), Fondecyt 1140090 (E.G.).

References

Aracena, J., Demongeot, J., Fanchon, E., and Montalva, M. (2013). On the number of different dynamics in boolean networks with deterministic update schedules. *Mathematical biosciences*, 242(2):188–194.

Goles, E., Montalva, M., and Ruz, G. A. (2013). Deconstruction and dynamical robustness of regulatory networks: Application to the yeast cell cycle networks. *Bulletin of mathematical biology*, 75(6):939–966.

Ruz, G. A., Goles, E., Montalva, M., and Fogel, G. B. (2014). Dynamical and topological robustness of the mammalian cell cycle network: A reverse engineering approach. *Biosystems*, 115:23–32.

Veliz-Cuba, A. and Stigler, B. (2011). Boolean models can explain bistability in the lac operon. *Journal of Computational Biology*, 18(6):783–794.

Discussion of synchronization problems during cell cycle in artificial cell modeling

Eugenia Schneider¹, Michael Mangold¹

¹Max-Planck-Institute for Dynamics of Complex Technical Systems, 39106 Magdeburg, Germany eschneider@mpi-magdeburg.mpg.de mangold@mpi-magdeburg.mpg.de

Extended abstract

In the last decades, progresses were made in the insight of microbiological processes. The mathematical description of biological processes by system biological models has been widely accepted as useful for a deepened understanding of existing biological systems. This development paves the way towards a systematic construction of artificial biological systems with tailored properties, which is the topic of synthetic biology.

Our aim is to evolve an artificial cell model consisting of functional biological devices like genome, transcriptome, proteome and metabolome. Although various mathematical models have been proposed to describe an artificial cell (Gánti, 2003; Novák and Tyson, 2008), there is a need for further theoretical analysis and mathematical modeling for a proper understanding of interactions between the functional devices.

Most of the artificial cell models may be structured into three functional devices representing a container forming the boundary of the cell, a metabolism generating the building blocks of the cell, and a programming part containing genetic information and regulating the processes inside the cell (Rasmussen et al., 2003). Self-replication of an artificial cell requires a synchronization of those three functional devices in such a way that at the end of the cell cycle the material in each of the devices has at least doubled. A key problem, which is addressed in this work, seems to be how to find a structure that guarantees this synchronization in a robust way, i.e. more or less independent of the kinetic parameter values. The need for the biological robustness is justified by the aim to be able to deal with perturbations.

As a starting point, we consider the Chemoton model described by T. Gánti (Gánti, 2003). The Chemoton consists of three self-reproducing functional devices: the autocatalytic chemical cycle representing the metabolism, the template polymerization subsystem serving as an information carrier and the membrane representing a container which grows proportional to the polymerization process. Although the Chemoton model is able to show self-sustained oscillations, these oscillations are not necessarily synchronized with cell

growth. To achieve this, a careful tuning of the kinetic parameters is required. The question is, if there is a model structure related to the Chemoton approach that possesses an inherent mechanism guaranteeing the synchronization of metabolism, program and container for a wide range of kinetic parameter values.

As a first step towards such a model structure, we combine the devices of the Chemoton model with the less complex structure of minimal cascade model for the mitotic oscillator described by Goldbeter (1991). The schematic representation of our artificial cell model is shown in Figure 1.

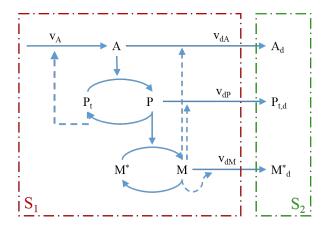


Figure 1: Artificial cell model based on Goldbeter's minimal cascade including the functional devices of Chemoton. System S_1 represents the mother cell and contains the metabolism A, the polymer template P_t , the new growing polymer P, catalytically inactive membrane building blocks M^* and catalytically active membrane building blocks M. The blue solid arrows display catalysis and the dashed arrows show induction. S_2 represent the daughter cell with metabolism component A_d , the polymer template $P_{t,d}$ and the membrane initially composed of inactive membrane building blocks M_d^* .

The mother cell consists of three main devices: the metabolism (A), the polymerization subsystem and the

membrane. We assume that the total quantity of the polymer template P_t and the growing polymer P is $P_t + P = 2$. The membrane is assumed to consist of catalytically inactive building blocks M^* and catalytically active building blocks M whose total concentration amounts to $M^* + M = 2$. The activity of metabolism A is induced by the polymer template $P_t = (2 - P)$ with a constant rate v_A and hence slows down towards the end of the polymer replication when there is hardly P_t in the cell. The polymer replication is catalyzed by A and terminates when P_t is consumed. During the polymerization the growing polymer P catalyzes the conversion of catalytically inactive membrane building blocks M^* into the active state M. After the terminated replication the double-stranded polymer splits into the polymer template P_t and the new polymer $P_{t,d}$ of the daughter cell induced by Mwith a constant rate v_{dP} . Furthermore, M triggers the transformation of A and M to daughter cell components A_d and M_d^* with constant degradation rates v_{dA} , v_{dM} . The conversion into the daughter cell devices initiates the cell division in our model. Obviously the proposed structure establishes a close interaction between the activation of the metabolism and the replication of the polymer. This interaction forces a synchronization between metabolism and polymerization. For the sake of simplicity volume changes of the mother cell are neglected in a first step. The mass balances of the components A, P, M, A_d , $P_{t,d}$ and M_d^* result in the following set of differential equations:

$$\begin{array}{rcl} \frac{dA}{dt} & = & v_A \left(2 - P \right) - \frac{v_{dA}MA}{K_{mdA} + A} \\ \frac{dP}{dt} & = & \frac{k_1 A \left(2 - P \right)}{K_{m1} + \left(2 - P \right)} - \frac{k_2 P}{K_{m2} + P} - v_{dP} MP \\ \frac{dM}{dt} & = & \frac{k_3 P \left(2 - M \right)}{K_{m3} + \left(2 - M \right)} - \frac{k_4 M}{K_{m4} + M} - v_{dm} M^2 \\ \frac{dA_d}{dt} & = & \frac{v_{dA} MA}{K_{mdA} + A} \\ \frac{dP_{t,d}}{dt} & = & v_{dP} MP \\ \frac{dM_d^*}{dt} & = & v_{dm} M^2 \end{array}$$

The model shows an oscillatory behavior with a stable limit cycle based on the model structure (Figure 2 a, b). When the activity of metabolism A increases, the replicating polymer P increases as well and only then the concentration of the catalytically active building blocks M increases. At high concentrations of M the concentrations of A and P begin to decrease caused by conversion into daughter cell components. As soon as the replicated polymer is split induced by M the metabolism A becomes active again. It is assumed that the cell division occurs at a time point t_{div} , when the concentration of M decreases caused by inactivation of M during the separation of the daughter cell. The concentrations A_d , $P_{t,d}$ and M_d^* at the time point t_{div} are

chosen to determine new initial conditions of the daughter cell: $A_d(t_0) = A_d(t_{div})$, $P_d(t_0) = 2 - P_{t,d}(t_{div})$, $M_d(t_0) = 2 - M_d^*(t_{div})$. The dynamic behavior simulation of the daughter cell is repeated with the new initial conditions. As shown in Figure 2 c, the daughter cell converges on a periodic cycle as well.

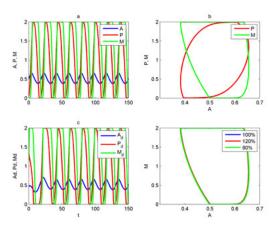


Figure 2: Simulation results of the artificial cell model: dynamic behavior of mother cell components A, P, M (a) and stable limit cycles (b); dynamic behavior of daughter cell's metabolism A_d , growing polymer P_d , catalytically active membrane building blocks M_d (c); limit cycles for different parameter sets of mother cell components (d).

The self-reproducing property of the model could be verified by consideration of daughter cell components. The quantities of conversed materials are sufficient to initiate a new cell cycle of the daughter cell after a certain phase of adaptation (Figure 2 c). Thus the model offers a robust self-replicating state. Furthermore, the system shows the same periodicity for parameter changes of $\pm 20\%$ (Figure 2 d) and is therefore robust in respect of parameter uncertainties. Further interesting aspects, which we want to consider, are how could we modify the structure of Chemoton model to achieve the same qualitative behavior as the system described here.

References

Gánti, T. (2003). Chemoton theory, volume 1: Theoretical foundations of fluid machineries, pages 144–159. Kluwer Academic/Plenum Publishers.

Goldbeter, A. (1991). A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase. *Proceedings of the National Academy of Sciences*, 88(20):9107–9111.

Novák, B. and Tyson, J. J. (2008). Design principles of biochemical oscillators. *Nature Reviews Molecular Cell Biology*, 9(12):981–991.

Rasmussen, S., Chen, L., Nilsson, M., and Abe, S. (2003). Bridging nonliving and living matter. *Artificial Life*, 9(3):269–316.

An enhanced artificial ecosystem: Investigating emergence of ecological niches

Morteza Mashayekhi¹, Abbas Golestani¹, Yasaman Majdabadi Farahani¹, and Robin Gras^{1,2,3}

¹ School of Computer Science, University of Windsor
² Department of Biology, University of Windsor
³ Great Lakes Institutes for Environmental Research, University of Windsor mashaye@uwindsor.ca

Abstract

This paper describes the new version of EcoSim, an individual-based predator-prey ecosystem simulation, using the updated 7-points Overview-Design concepts-Details (ODD) standard protocol for describing the individual-based models. New sensing and action concepts have been added to the fuzzy cognitive map (FCM) of the individuals. In addition, new physical traits have been added to the current genome allowing different niches to emerge. Experimental results demonstrated that different ecological niches emerged with a co-evolution of highly differentiated behavior and physical traits.

Introduction

Among biological disciplines, behavioral ecology has a strong tradition of accounting for the role of organism-environment interactions in behavior [1]. Behavioral ecology and the related field of optimal foraging theory [2] model animal behavior in terms of optimal adaptation to environmental niches. The goal is not to test whether organisms actually behave optimally, but to use normative expectations to interpret behavioral data and/or generate testable hypotheses. One approach for understanding the behavior of complex ecosystems is individual-based modeling which provides a bottom-up approach allowing for the consideration of the traits and behavior of individual organisms. Ecological modeling is still a growing field, at the crossroad between theoretical ecology, mathematics and computer science [3]. Since natural ecosystems are very complex (in terms of number of species and of ecological interactions), ecosystem models aim to characterize the major dynamics of ecosystems. in order to synthesize the understanding of such systems, and to allow predictions of their behavior. Ecosystem simulations also can help scientists to understand theoretical questions regarding the evolutionary process, the emergence of species, and the emergence of learning capacities. One of the main interests of such ecosystem simulations is that they offer a global view of the evolution of the system, which is difficult to observe in nature. However, the scope of ecosystem simulations has always been limited by the computational possibilities of their time. Today, it is possible to run simulations that are more complex than what has ever been done before, due to the available high performance computing resources.

There exists several ecosystem simulation platforms with various features. For example, Echo is one of the first such ecosystem models. It is a "genetic ecosystem model in which evolving agents are simulated in a resource-limited environment" [4]. In Echo, each agent, when obtaining the required resources to copy its genome, replicates itself with some mutations. The agents, by interaction with other agents (combat, trade, or mating) or from the environment, can acquire resources. Polyworld is another software developed by Larry Yaeger [5] to evolve artificial intelligence through natural selection and evolutionary algorithms. It displays a graphical environment in which a population of trapezoid agents search for food, mate, and create offspring. The population is typically only in the hundreds, as each individual is rather complex and the environment consumes considerable computer resources. In this model, each individual makes decisions based on a neural network which is derived from each individual's genome. Avida is another artificial life software platform to study the evolutionary biology of self-replicating and evolving computer programs (digital organisms) [6], which was inspired by the Tierra system [7]. Unlike Tierra, Avida assigns every digital organism its own protected region of memory, and executes it with a separate virtual CPU. A second major difference is that the virtual CPUs of different organisms can run at different speeds. The speed at which a virtual CPU runs, is determined by a number of factors, but most importantly, by the tasks that the organism performs: logical computations that the organisms can carry out to reap extra CPU speed as a bonus.

EcoSim [8] is a large scale evolving predator-prey ecosystem simulation that can be used to perform studies in theoretical biology and ecology [9, 10]. It has been shown that EcoSim generate patterns as complex as those observed in real ecosystems [11]. Several studies have been done using EcoSim. Devaurs and Gras have shown that the behavior of this model is realistic by comparing the species abundance patterns observed in the simulation with real communities of species [12]. Furthermore, the chaotic behavior [11] with multi-fractal property [13] of the system has been proved as the ones observed in real ecosystems [14] and Golestani et al. have measured the effect of small geographic barriers on the speciation in EcoSim [9]. The effectiveness of the spatial distribution of individuals in the speciation has been investigated by Mashayekhi and Gras [15]. Marwa et al. demonstrated that introduction and predator removal from an

ecosystem have widespread effects on the survival and evolution of prey by altering their genomes and behavior [16]. Finally, Mashayekhi et al. proved that the extinction mechanisms in EcoSim are similar to those of real communities [10].

Ecological niches describe the way of life of a species which is unique to that species. They explain how an organism or a population responds to the distribution of resources, competitors, and predator. For example, one species may increase its size when the resources are abundant or when the predators are scarce, or one species can run faster to avoid the competitors or predators. In fact, niches describe the species' role or function within the community and how it adapts to life in its habitat.

In addition to presenting the new version of EcoSim following the updated 7-points Overview-Design concepts-Details (ODD) standard protocol [17], we investigate the effect of physical traits in forming ecological niches in EcoSim. The paper is organized as follows: in the next section, we present the ODD description of the modified version of EcoSim. Thereafter, we present the results of analyses of population and diversity of species of the simulation.

ODD Description of EcoSim

EcoSim is an individual-based ecosystem simulation, designed to simulate agents' behavior in a dynamic, evolving ecosystem. The agents (or individuals) of EcoSim are prey and predators acting in a simulated environment. In order to study the effect of evolution of physical traits, several features have been added to the current version of EcoSim such as: new individuals' perceptions of their environment, new actions, and new physical traits, which we call the physical genome. The ODD description [10] of the modified version of EcoSim is given below.

Purpose

EcoSim was designed to simulate agents' behavior in a dynamic and evolving ecosystem. The main purpose of EcoSim is to study biological and ecological theories and construct a complex adaptive system which leads to a generic ecosystem platform with behaviors similar to those found in existing ecosystems. Due to the complexity, scale, and resource requirement of studying these theories in real biological systems, simulations of this nature are quite necessary. EcoSim uses a fuzzy cognitive map (FCM) [18] to model the agent's behavior. The agent's FCM, being coded in its genome, allows the evolution of the agent behavior through the epochs of the simulation.

Entities, state variables, and scales

Individuals. EcoSim has two types of individuals: predator and prey. Each individual possesses two types of traits: acquired and inherited traits (Table 1). The former varies depending on the environmental conditions and the latter is encoded in the individual's genome and is fixed during its lifetime. The age and speed are initialized to zero for new born individuals while energy, a crucial property of the

individual, is initialized based on the amount of the parents' energy which is invested at the breeding time (*State of Birth* or *SOB*). Afterward, energy is provided to the individuals by the resources (food) they find in their environment. Prey consume grass, which is dynamic in quantity and location, whereas predators hunts for prey individuals. Strength of the individual is calculated based on the current energy, maximum energy, and age of the individual. Young and old individuals have less strength.

The genome of each individual consists of two parts: a physical genome that represents the physical characteristics of the individual and the FCM, which form its behavioral model.

Table 1: Several physical and life history characteristics of individuals from 5 independent runs (The values for the inherited features are the values at initialization and for the acquired features, they are the average values over 20000 time steps)

Type	Characteristic	Predator	Prey
	Maximum Energy	800 units	650 units
	Maximum Age	42 time steps (±6)	46 time steps (±18)
	Vision	20 cells maximum	13 cells maximum
Inherited	Maximum Speed	11 cells / time step	6 cells / time step
ĮUĮ	Minimum age of reproduction	6 time steps	6 time steps
	State of Birth	30	30
	Defense	N/A	0
	Cooperative Defense	N/A	0
Acquired	Average Energy	753(±175)	804(±275)
	Average Age	17(±3.4)	19(±4.5)
	Average speed	3 (±0.8)	9.4(±2.6)
1	Average Strength	1562(±395.5)	1275(±376)

Each individual performs one unique action during a time step, based on its perception of the environment. At each time step, each agent spends energy depending on the selected action (e.g. breeding, eating, running), and also on the complexity of its behavioral model (number of existing edges in its FCM) as well as its physical genome. (See equations 1, energy penalty for prey, and equation 2, energy penalty for the predator)

$$P = \frac{FC - 100}{3.1} + S^{1.5} + (\frac{ME}{18})^{1.25} + (\frac{V}{4})^{2} + (\frac{MS}{5})^{1.5} + (\frac{D}{18})^{1.7} + (\frac{CD}{18})^{1.7} + (max(0, 7 - RA))^{2.1}$$

$$P = \frac{FC - 100}{3.1} + S^{1.5} + \left(\frac{ME}{15}\right)^{1.4} + \left(\frac{V}{4}\right)^{1.9} + \left(\frac{MS}{5}\right)^{1.5} + \left(max(0,7 - RA)\right)^{2.3}$$

In these equations, FC is the number of edges in the FCM, S is the current speed, ME is the maximum energy, V is the vision range, MS is maximum speed, and RA is the minimum reproduction age. D and CD stand for defense and cooperative defense, respectively.

Cells and virtual world. The smallest units of the environment are cells. Each cell represents a large space which may contain an unlimited number of individuals and/or some limited amount of food. The virtual world consists of a matrix of 1000×1000 cells. The world is large enough such that an individual moving in the same direction during its whole life cannot even cross half of the world and therefore migration patterns can be observed,. The virtual world wraps around to remove any spatial bias. In addition, the dimensions of the world are adjustable but expanding the dimensions increases the computational complexity of the simulation.

Time step. Each time step involves the time needed for each agent to perceive its environment, to make a decision, to perform its action, to update the species membership, to perform speciation events, and to record relevant parameters (e.g. the quantity of available food). In terms of computational time, the speed of simulation per time step is related to the number of individuals. Recent executions of the simulation produced approximately 20000 time steps in 90 days.

Population and Species. At almost every time step several prey and predator species co-exist. A species is a set of individuals with similar genomes (see 'collectives in design concepts' section for more details).

Entities, state variables, and scales

After perceiving its environment (including grass resources, predators, and sexual partner), the possible actions for a prey agent are: evasion (escape from predator), search for food (if there is not enough grass available in its cell, prey can move to another cell to find grass), socialization (moving to the closest prey in the vicinity, moving to the cell with strongest prey, moving to the cell with maximum total prey's strength, and moving to a cell with minimum total prey's strength), exploration, resting (to save energy), eating, and breeding. Predators also perceive the environment to gather information used to choose an action among: hunting (to catch a prey), moving to the cell with strongest prey, moving to the cell with minimum total prey's strength, moving to the cell with the weakest prey, search for food, socialization (moving to the closest predator in the vicinity, moving to the cell with strongest predator), exploration, resting, eating, and breeding. Every individual takes one action per time step then its energy level and its strength are adjusted. The age of all individuals are also increased by one unit at each time step. In addition, there are two environmental processes which are adjusted: the amount of grass and meat after all the individuals perform their actions.

At each time step, the value of the state variables of individuals and cells are updated. The overview and scheduling of every time step is as follows:

- 1. For prey individuals:
 - 1.1. Perception of the environment
 - 1.2. Computation of the next action
 - 1.3. Performing of their action and updating of their energy and strength

- 1.4. Updating the list of prey
- 1.5. Sorting prey based on the strength
- 1.6. Updating prey species
- 2. For predator individuals:
 - 2.1. Perception of the environment
 - 2.2. Computation of the next action
 - 2.3. Performing of their action and updating of their energy and strength
 - 2.4. Updating the list of predators and prey
 - 2.5. Sorting predators based on the strength
 - 2.6. Updating predator species
- 3. For every cell in the world
 - 3.1 Updating of the grass level
 - 3.2 Updating of the meat level
- 4. Updating of the age of the individuals

The complexity of the simulation algorithm is mostly linear with respect to the number of individuals. If we consider that there are N_I prey and N_2 predators and we exclude the sorting parts which have a complexity of $O(N_I log N_I)$ and $O(N_2 log N_2)$ but are negligible in computational time, then the complexity of part 1 and part 2 of the above algorithm, including the clustering algorithm used for speciation, will be $O(N_I)$ and $O(N_2)$ respectively (Aspinall and Gras, 2010). This virtual world of the simulation has 1000×1000 cells, therefore the complexity of part 3 will be $O(k = 1000 \times 1000)$. The complexity of part 4 will be $O(N_I + N_2)$. As a result, the overall complexity of the algorithm is $O(2N_I + 2N_2 + k)$, which is $O(N_I + 2N_I)$.

Design concepts

Basic principles

A FCM is the base for describing and computing the agents' behaviors [8]. Agents act based on their FCM to decide their next action. Their FCM is represented as part of their genome which is assigned to each individual at birth. A FCM is a directed graph containing nodes representing concepts and edges representing the influence of concepts on each other [18]. When a new offspring is created, it is given a genome which is a combination of the genomes of its parents with some possible mutations. The behavior model of each individual is therefore unique. Formally, a FCM is a graph which contains a set of nodes C (each node C_i is a concept) and a set of edges I (each edge I_{ii} representing the influence of the concept C_i on the concept C_i). A positive weight associated with the edge I_{ii} corresponds to an excitation of the concept C_i from the concept C_i , whereas a negative weight is related to an inhibition (a zero value indicates that there is no influence of C_i on C_i). The influence of the concepts in the FCM can be represented by a $n \times n$ matrix, L, in which n is the number of concepts and L_{ij} is the influence of the concept C_i on the concept C_i . If $L_{ij} = 0$, there is no edge between C_i and C_{i} .

Emergence

In each FCM, three kinds of concepts are defined: sensory (such as distance to foe or food, amount of energy, etc.), internal (fear, hunger, curiosity, satisfaction, etc.), and motor or action (evasion, socialization, exploration, breeding, etc.). The activation level of a sensory concept is computed by performing a fuzzification of the information the individual

perceives in the environment. For an internal or motor concept C, the activation level is computed by applying the defuzzification function on the weighted sum of the current activation level of all the concepts having an edge directed toward C. Finally, the action of an individual is selected based on the maximum value of motor concepts' activation level. Activation levels of the motor concepts are used to determine the next action of the individual.

At the initiation of the simulation, prey and predators are scattered randomly all around the virtual world. Through the course of the simulation, distribution of the individuals in the world is changed drastically based on many different factors such as prey escaping from predators, individuals socializing to form groups, and individuals migrating to find food resources. The complex interactions between the individuals and their environment lead to emerging grouping patterns with spiral waves. Individuals' distribution forming spiral waves is one property of prey-predator models [19]. In addition, migration phenomena can be observed due to the combination of the predation pressure, and the search for resources and potential mates.

Adaptation

The FCM maximal length is fixed (663 genes for prey and 756 for predator), where each site corresponds to an edge between two concepts of the FCM. However, as many edges have an initial value of zero, only 117 edges for prey and 131 edges for predators exist at initialization. In addition to the FCM, every individual has physical traits (or physical genome) in order to describe its physical characteristics, each trait being coded by one gene. Maximum energy, maximum age, vision, maximum speed, and state of birth are common physical traits for prey and predator. Prey has also two more traits: defense, and cooperative defense, to be able to protect themselves of predators. The initial values for these two genes are zero. Every prey can have its own defense capability and also can participate in a cooperative defense with other prey against predators. When one individual participates in a defense position, it loses some amount of energy. Predators also have an energy penalty when it encounters a prey with defense ability or a group of prey with cooperative defense capability. It is even possible for a predator to be killed in this process. Step after step, as more individuals are created, changes in the FCM and physical genome occur due to the formation of new edges (with probability of 0.001), removal of existing edges (with probability of 0.0005), and changes in the weights associated to existing edges (with probability of 0.005). New genes may emerge from the initial pool of edges with a zero value. This emergence and disappearance of the genes in FCM is due to natural selection and genetic drift which lead to adaptability of individuals [20].

Fitness

A species' fitness is calculated as the average fitness of its individuals. The fitness of an individual is defined as the age of death of the individual plus the sum of the age of death of its direct offspring. Accordingly, the fitness value mirrors the individual's capability to survive longer and produce high number of strong adaptive offspring. There is no pre-defined explicit fitness-seeking process in the simulation, but rather it is a consequence of natural selection. Individuals which are more adapted to the environment live longer, have a higher

level of energy, and therefore are able to have more offspring and can transfer efficient genomes to them.

Prediction

So far, there is no learning mechanism for individuals and they cannot predict the consequences of their decision. The only available information, for every individual to make decision with, is the information coming from their perceptions at the current time step and the value of the activation level of the internal and motor concepts at the previous time steps. The activation levels of the concepts of an individual are never reset during its life. As the previous time step activation level of a concept is involved in the computation of its next activation level, this means that the previous states of an individual participate in the computation of its current state. Therefore, an individual has a basic memory of its own past that will influence its future states. As the action undertaken by an individual at a given time step depends on the current activation level of the motor concepts, the global behavior of the individual depends on a complex combination of the individual's perception, the current internal states, and the past states it went through during its life.

Sensing

Every individual in EcoSim is able to sense the local environment inside of its vision range. Some of these sense abilities are common between prey and predator. For instance, both can perceive information about cells with food units and the number of likely mates within the vision range, current level of energy, age, and strength. In addition, both sense the strength of the strongest prey in the same cell, the sum of the strength of the prey in the local cell, and the strongest sum of the strength of the prey in the vision range cells. Moreover, predators can sense the closest prey and the strength of the strongest predator in the local cell.

Interaction

The only action that requires a coordinated decision of two individuals is reproduction. In order to have successful reproduction, the two parents need to be in the same cell, have enough energy, choose the reproduction action, and be genetically similar. The individuals cannot determine their genetic similarity with their potential partner. They try to mate and if the partner is too dissimilar, (the dissimilarity between the two genomes is greater than a threshold i.e. half of the speciation threshold), the reproduction fails.

Predator's hunting introduces another type of interaction in the simulation. For a predator to succeed in the hunting action, the target prey should be in the same cell as the predator is located.

Furthermore, there is a competition for prey and predators for food. For example, if in a given cell two agents choose the action of eating, the more powerful agent (in terms of strength) has more priority and acts first.

Stochasticity

To produce variability in the ecosystem simulation, several processes include stochasticity. For instance, at initialization time the amount of grass units is randomly determined for each cell (a value between 1 and *MaxGrass*). Moreover, the maximum age of an individual is determined randomly at birth from a uniform distribution centered at a value associated with its maximum value coming from the trait

"maximum age" (MaxAge). Stochasticity is also included in several actions of the individuals; in evasion and socialization: if there is no predator or partner in the vision range of the individual, the direction of the movement would be random. Furthermore, the direction of the exploration action is always random.

Collectives

In EcoSim, the notion of species is implemented in a way that species emerge from the evolving population of agents. Species can become extinct if all of their members die. EcoSim implements a species based on the genotypic cluster definition [21] in which a species is a set of individuals sharing a high level of genomic similarity. In addition, in EcoSim, each species is associated with the average of the genetic characteristics of its members, called the 'species genome' or the 'species center'. The speciation mechanism implemented in EcoSim is based on the gradual divergence of individual genomes. The speciation method begins by finding the individual A in a species S with the greatest distance from the species center. The distance is the sum of the hamming distance between the physical genomes and between the FCM edges. Next, the individual B in S with the greatest distance to A is found. If this distance is greater than a predefined threshold for speciation, a 2-means clustering is performed [22], otherwise S stays unchanged.

To initialize the 2-means clustering process, one center is assigned to a random individual, denoted I_r , and the other center is assigned to the individual which is the most genetically different from I_r . After, eight cycles of the 2-means clustering algorithm, two new sister species are created to replace S.

Observation

EcoSim produces a large amount of data at every time step, including new and extinct species, geographical and internal characteristics of every individual, and status of the cells of the virtual world. Information regarding each individual includes position, level of energy, choice of action, species, parents, FCM, etc. Information about the individuals, species, and virtual world for every 100 time steps are stored in a file using HDF5 format [23] with an average size of 6 GB. Also there is a possibility to store all of the values of every variable in the current state of the simulation in a separate file, giving the possibility to restore the simulation from that state afterwards. The overall size of this file, which is only stored in every 100 time steps of the simulation, is a few GBs depending on the size of population and species. All the data is stored in a compact special format, to facilitate the storage and future analysis. There are also several program utilities which can be used to analyze the simulation outputs for example, to calculate the species and individuals fitness, to demonstrate the world snapshot for each time step of the simulation, generate the video of the world, and to draw the FCM of the individuals.

Initialization and input data

A parameter file is defined for EcoSim, which is used to assign the values for each state variable at initial time of the simulation. These parameters are as follows: width and height of the world, initial numbers of individuals, thresholds of genetic distance for prey/predator speciation, initial maximum

age, energy, speed, vision range, and initial values of FCM for prey/predator. Any of these parameters can be changed for specific experiments and scenarios. An example of a list of most common user specified parameters for initially running the EcoSim are presented in Table 2.

Table 2. Values for user specified parameters.

User Specified Parameter	Used Value
Number of Prey	12000
Number of Predators	9000
Max Grass Quantity in each cell	1600
Prey Energy	650
Predator Energy	800
Prey Vision Range	13
Predator Vision Range	20

Submodels

At initialization, there is no meat in the world and the number of grass energy units is randomly determined for each cell. In this case, the initial number is generated uniformly between 1 and MaxGrass. When a prey eats, the obtained energy is equal to min(MaxEnergy - CurrentEnergy, GrassUnits), which GrassUnits is the number of grass energy units, and then the amount of grass in the cell is decreased by this value. Predators have two modes of predation: hunting and scavenging. When a predator's hunting action succeeds, some amount of meat energy units, equal to the strength of the killed prey, is added to the cell. Afterward, the predator consumes the meat to gain required energy i.e. min(MaxEnergy - CurrentEnergy, MeatUnits), which MeatUnits is the number of meat energy units. The eventual remaining units of meat energy can be consumed by other predators. Prey have a defence capability as well as cooperative defence and use them in a battle against the predator [24]. Depending on the strength and defence ability of the attacked prev and the overall strength and cooperative defence ability of the other prey in the same cell, the predator may lose energy (See equation 3, AP.D and AP.S are defence and the strength of the attached prey. CP_iD and CP_iCD are the defence and cooperative defence ability of the prey i in the

$$P = A P . D * A P . S + \sum_{i} C P_{i} . D * C P_{i} . C D$$
 (3)

In addition, the prey which are involved in a cooperative defence also lose some amount of energy based on the strength of the predator (0.2 * *PredatorStrength*/ Number of defenders).

There are dynamic processes for the resources in each cell such as the amount of grass growth, grass diffusion, and variation in the amount of meat at each time step. The amount of meat in each cell increases by the strength of the dead prey in that cell. It also decreases at each time step, even if no meat has been eaten in this cell.

Each action also has its corresponding sub-model:

- 1. Evasion (for prey only). The evasion direction is the direction opposite to the barycenter of the five closest predators within the vision range of the prey, with respect to the current position of the prey. If no predator is within the vision range of the prey, the direction is chosen randomly. The current activation level of the *fear* concept is divided by two after evasion.
- 2. *Hunting* (for Predator only). The predator selects the closest cell (including its current cell) that contains at least one prey and moves toward that cell. If it reaches the corresponding cell based on its speed, the predator tries to kill a prey as explained before. When there are several prey in the destination cell, one of them is chosen randomly. If the speed of the predator is not enough to reach the prey, it moves at its speed toward this prey. If there is no prey in the current cell and in the vicinity hunting action is failed.
- 3. Search for food. The direction toward the closest food (grass or meat) within the vision range is computed. If the speed of the agent is high enough to reach the food, the agent is placed on the cell containing this food. Otherwise, the agent moves at its speed toward this food.
- 4. Socialization. The direction toward the closest possible mate within the vision range is computed. If the speed of the agent is high enough to reach the mate, the agent is placed on the cell containing this mate, and the current activation level of the SearchPartner concept is divided by three. Otherwise, the agent moves at its speed toward this mate. If no mate is within the vision range of the agent, the direction is chosen randomly.
- 5. *Exploration*. The direction is computed randomly. The agent moves at its speed in this direction. The activation level of the *curiosity* concept is divided by 1.5 after exploration action.
- 6. Resting. Nothing happens.
- 7. *Eating*. If the current amount of grass (of meat) is greater than 0, then this number is decreased by the required energy and the prey's (predator's) energy level is increased as discussed earlier. Its activation level for the *hunger* concept is divided by four.
- 8. Breeding. The process of generating a new offspring consists of the following steps. First, the edges' values in the FCM along with the values in physical genome from one randomly chosen parent are transmitted with possible mutations to the offspring. Then, the initial energy of the offspring is computed based on the parents SOB and the mutated MaxEnergy. To model the crossover mechanism, the edges are transmitted by block from one parent to the offspring. For each concept, its incident edges' values are transmitted together from the same randomly chosen parent. Finally, the energy level of the two parents is decreased by half of the energy transmitted to the offspring.
- 9. MovetoStrongestIndividual. The direction toward the strongest possible mate within the vision range is computed. If the speed of the agent is high enough to reach the mate, the agent is placed on the cell containing this mate, and the current activation level of SearchPartner is divided by three. Otherwise, the agent moves at its speed toward this mate. If no mate is within the vision range of the agent, the direction is chosen randomly.
- 10. Move2StrongestPreyCell (for prey only). This action follows the MovetoStrongestIndividual action's steps except

- that the direction of movement is toward the cell with the highest cumulative strength of prey individuals to allow prey to benefit from cooperative defence against predators.
- 11. Move2WeakestPreyCell (for prey only). This action is similar to Move2StrongestPreyCell, but the direction of movement is toward the cell with the lowest cumulative strength of prey individuals to allow prey to have higher chance in competition with other prey individuals for accessing to food.
- 12. Move2StrongestPreyDistance (for predator only). The predator moves toward the strongest prey individual to acquire more energy after possible hunting. If the speed of the agent is high enough to reach the prey, the agent is placed on the cell containing this prey, and the current activation level of ChaseAway is divided by two. If the speed of the predator is not enough to reach the prey, it moves at its speed toward this prey.
- 13. *Move2WeakestPrey* (for predator only). This action has the same steps of the *Move2StrongestPreyDistance* action with the exception of the direction of movement is toward the weakest prey individual for easy hunting in the next steps.
- 14. Move2WeakestPreyCell (for predator only). This action is similar to Move2WeakestPrey, but the direction of movement is toward the cell with the lowest cumulative strength of prey individuals to minimize the possible effect of cooperative defence by prey individuals.

For all movement actions M the speed is equal to MaxSpeed*ActivationLevel(<math>M).

Results and Discussion

Population and Species

Figure 1 shows the average population size for the prey and predators for 20000 time steps of five different runs of the simulation. As we expected, there is a dependency between prey and predator populations. At the beginning of the simulation, the number of prey is more than the number of predators. Therefore, they can reproduce and increase their population very fast. The increasing number of prey provides a good chance for the predators to have access to more food resulting in an increasing of their energy level and their reproduction rate. The resulting increase in hunting associated with a food resource shortage for prey decreases the prey population gradually and subsequently the predator population also declines. This cycle happens several times during the simulation with a time lag between the prey and predator population fluctuations. Both populations mostly stabilize with small fluctuations at the end of simulation, resulting in a prey population around 200000 and predator about 30000 individuals. The number of species for both prey and predators are positively correlated with the population size. An increase in the population size increases the variation in the gene pool and consequently increases the chance of the occurrence of a new species. This fact is shown in figure 2.

Physical Traits

With the current energy formula, individuals tend to increase their size (MaxEnergy) and their MaxAge. The individuals' visions also tend to decrease generally, but we observed that prey individuals prefer to have less vision range compared to predator. On the other hand, prey individuals have tendency to

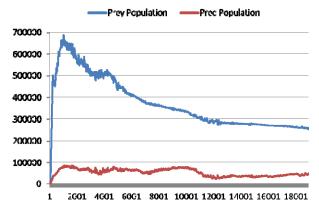


Figure 1 Average population size of 5 different runs for 20000 time steps

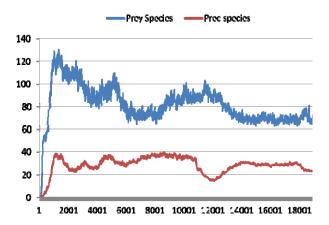


Figure 2 Average number of species of 5 different runs increase their *MaxSpeed* to the maximum possible value while it stays almost fixed for predators (figure 3). While it is clear that the individuals try to become larger in order to be able to gain more energy and strength, it is less obvious why the *MaxAge* increases despite its cost in energy. There is no predefined fitness function giving preference to longer life expectancy and number of offspring, but it could represent a long term advantage for the species by increasing the overall birth ratio.

Defense and cooperative defense abilities of prey individuals, which are set to zero at initialization of the simulation, are also evolving. We observed that these abilities are used by the prey during some periods, but sometimes they decrease because of their associated energy penalty. However, the general trend for these abilities is increasing gradually.

Actions

We investigated the global behavior of different species in our system such as the average percentage of species' actions that can give insight into the decision-making processes of species to see if some separate, unique niches emerged. By monitoring the actions performed by predator individuals in different species, we can study how an individual or species responds to the distribution of resources and competitors, and how it consequently alters those same factors.

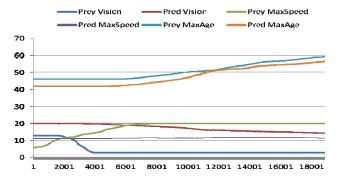
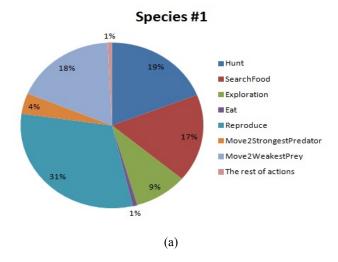
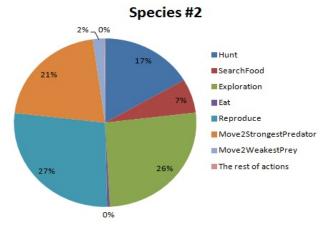


Figure 3 Evolution of some physical genome traits





(b) Figure 4 Percentage of different actions for two different species.

Our results clearly show how natural selection drives competing species into different patterns of resource usage. Figure 4 shows the average amount of actions chosen by predator individuals of two different species. Aside from reproduction which is an important action in both species, the percentage of other actions is different. For example species #1 has high amount of movement toward the weakest prey in order to increase the chance of success in predation. It seems that this policy alongside high amount of search for food

action works for this species in order to survive in the ecosystem. On the other hand, species #2 has a high amount of exploration and movement toward the strongest partner. These actions increase the ability of producing fit offspring for the next time steps.

Various physical traits evolve differently in these two species which is further evidence of niche adaptation. Some of these differences may be in direct relation with their difference in behavior (see figure 4 and figure 5). For example, species #1 has a lower MaxEnergy and consequently a lower strength than species #2 which can explain that they prefer scavenging (SearchFood) and selecting the weakest prey. Conversely, species #2 being more efficient in hunting can dedicate more of their energy to reproduction and selection of a strong mate.

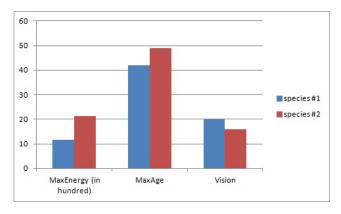


Figure 5 Physical trait values in two different species.

Conclusion

We added new sensing as well as new action concepts to the fuzzy cognitive map (FCM) of the individuals in EcoSim. In addition, new physical traits have been added to the behavioral genome allowing different niches to emerge. Our results underline the importance of ecological niches in evolution. Significantly, our study reveals insights into the genetic mechanisms of niche adaptation, advances our understanding of the evolution, and sheds more light on addressing more complicated biological questions. Being at the early stage of the analysis of the new version of EcoSim potentials, these preliminary results are promising and will lead to some more dedicated study on niche emergence.

Acknowledgements

This work is supported by CRC grant 950-2-3617 and the CFI grant 203617 and is made possible by the dedicated resource allocation 8047 of the Shared Hierarchical Academic Research Computing Network (SHARCNET, www.sharcnet.ca). We also appreciate Ryan Scott for his helpful comments.

References

[1] Krebs, J. and Davies, N. (1997) Behavioural ecology: an evolutionary approach, 4th edn. Oxford, UK: Blackwell Publishers.

- [2] Stephens, D. and Krebs, J. (1986) Foraging theory. Princeton, NJ: Princeton University Press.
- [3] Ricotta, C. (2000). From theoretical ecology to statistical physics and back: self-similar landscape metrics as a synthesis of ecological diversity and geometrical complexity. *Ecological Modeling*, 125(2-3):245–253.
- [4] Hraber, P. T., Jones, T., and Forrest, S. (1997). The ecology of echo. *Artificial Life*, 3(3):165–90.
- [5] Yaeger, L. (1994). Poly world: Life in a new context. Proc. Artificial Life III, pages 263-263.
- [6] Ofria, C. and Wilke, C. O. (2004). Avida: a software platform for research in computational evolutionary biology. *Artificial Life*, 10(2):191–229.
- [7] Thearling, K., Ray, T. (1994) Evolving multi-cellular artificial life, Artif. Life IV, vol. IV, pages 283–288.
- [8] Gras, R., Devaurs, D., Wozniak, A., and Aspinall, A. (2009) An individual-based evolving predator-prey ecosystem simulation using a fuzzy cognitive map as the behavior model, *Artificial Life*, 15(4):423–463.
- [9] Golestani, A., Gras, R., and Cristescu, M. (2012) Speciation with gene flow in a heterogeneous virtual world: can physical obstacles accelerate speciation?, *Proceedings of the Royal Society B: Biological Sciences*.
- [10] Mashayekhi M., MacPherson B., and Gras R. (2014) A machine learning approach to investigate the reasons behind species extinction, *Ecological Informatics*, 20:58-66.
- [11] Golestani, A., and Gras, R. (2010). Regularity analysis of an individual-based ecosystem simulation. *Chaos*, 20(4):3120.
- [12] Devaurs, D.and Gras R. (2010) Species abundance patterns in an ecosystem simulation studied through Fisher's logseries, Simulation Modeling Practice and Theory, 18(1):100–123.
- [13] Golestani A. and Gras, R. (2011) Multifractal phenomena in EcoSim, a large scale individual-based ecosystem simulation, *Int. Conf. Artificial Intelligence, Las Vegas*, pages 991–999.
- [14] Seuront, L., Schmitt, F., Lagadeuc, Y., Schertzer, D., Lovejoy, S., and Frontier, S. (1996) Multifractal analysis of phytoplankton biomass and temperature in the ocean, *Geophysical Research Letters*, 23(24):3591–3594.
- [15] Mashayekhi M. and Gras R. (2012) Investigating the effect of spatial distribution and spatiotemporal information on speciation using individual-based ecosystem simulation, GSTF Journal of Computing, 2(1):98–103.
- [16] Khater, M., Murariu, D., and Gras, R. (2014) Contemporary evolution and genetic change of prey as a response to predator removal, *Ecological Informatics*, 22:13-22.
- [17] Grimm, V., Berger, U., Bastiansen, F., Eliassen, S., Ginot, V., Giske, J., ... DeAngelis, D. L. (2006). A standard protocol for describing individual-based and agent-based models. *Ecological Modeling*, 198(1-2), 115–126.
- [18] Kosko, B. (1986). Fuzzy cognitive maps. International Journal of Man-Machine Studies, 24:65-75.
- [19] Golestani A. and Gras R., Identifying origin of self-similarity in EcoSim, an individual-based ecosystem simulation, using wavelet-based multifractal analysis, Proceedings of the world congress on engineering and computer science 2012 (WCECS 2012), San Francisco, 1275-1282.
- [20] Gras, R., Golestani, A., Cristescu M., and Hendry, A. P. (2014) Speciation without pre-defined fitness functions, *Science*, (submitted).
- [21] Mallet, J. (1995) A species definition for the modern synthesis. Trends in Ecology and Evolution, 10(7): 294-299.
- [22] Aspinall A. and Gras R. (2010) K-Means clustering as a speciation method within an individual-based evolving predator-prey ecosystem simulation, 6th Int. Conf. on Active media technology, pages 318-329, Springer-Verlag Berlin, Heidelberg.
- [23] The HDF Group. Hierarchical data format version 5, 2000-2014. http://www.hdfgroup.org/HDF5.
- [24] Arnold, K. E. (2000) Group mobbing behavior and nest defense in a cooperatively breeding Australian Bird, Ethology 106 (5): 385–393.

Chaining Distinct Tasks Drives The Evolution of Modularity

Brett Calcott¹

¹Center for Advanced Modeling, Johns Hopkins University, MD 21209 brett.calcott@gmail.com

Abstract

I introduce a novel method for evolving modularity in gene regulatory networks. Like previous models of modular evolution, it relies on selecting networks to perform a task consisting of distinct sub-tasks, with the aim of producing modules that perform these sub-tasks. Whilst existing models structure these sub-tasks in parallel and then combine the output, this model chains sub-tasks together, so they must be performed one after another. This task structure resembles the selective constraints undergone in multicellular evolution, where genetic networks must (a) integrate multiple cues to establish what environment they are in, and (b) express a pattern of gene activity on the basis of this environment. I show that the modules produced in these networks exhibit the hallmarks of modularity: existing modules can change independently of others, and modules can also be re-used and combined in various ways.

Extended Abstract

Modularity makes complex systems evolvable—it permits localised changes to be made without impacting the entire system, and it allows already functioning parts in a system to be re-used in novel ways. An important goal then, is to understand the conditions under which modularity can evolve. One way this problem has been explored is through simple models of network evolution, where nodes in the network are thought of as neurons or regulatory genes that compute some function. These nodes are then wired together into a network whose overall behaviour can be evaluated for how close it comes to some target behaviour (such as the computation of a boolean function).

Two recent network models that successfully evolve modularity share a central idea: networks are selected to solve a *modular task*—one that can be broken down into two parallel sub-tasks whose outputs are then combined into a single output (see Figure 1(b)) (Kashtan and Alon, 2005; Clune et al., 2013). When modularity in the network does evolve, it is possible to isolate individual modules that perform the separate sub-tasks. In both these models, additional constraints are required for the networks to reliably evolve modularity. Kashtan and Alon vary the sub-tasks rapidly and independently over time, whilst Clune et al. add a fitness cost

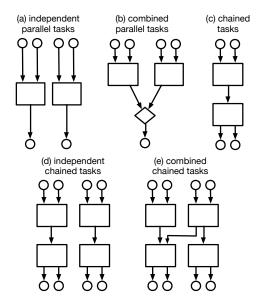


Figure 1: Different modular task definitions that networks can be selected to perform. Successful modular evolution should reflect a similar network structure. Previous models focus on (b). This method first develops (c) as a building block, and then uses it to evolve (d) and (e). Note that chained tasks must coordinate *multiple outputs*.

to the number of connections in network, demonstrating that both the external selective regime, and costs imposed by internal structure can affect the evolution of modularity.

I explore a novel method for evolving modularity in an existing model of gene regulation (Calcott et al., 2008) that similarly relies on selecting networks to perform a modular task. In this model, however, the sub-tasks are chained together, and must be performed one after another, rather than in parallel (see Figure 1 (c)). The upstream sub-task integrates information from multiple inputs into a single output, and the downstream sub-task uses that information to coordinate several outputs.

I show that this selective regime *alone* is often sufficient to evolve modules in the network that map onto the sub-tasks

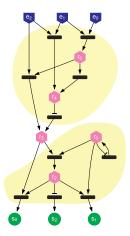


Figure 2: A modular network. Genes are black; environmental cues E_i , regulatory products R_j , and structural output S_k , are colored), showing a positional module and a structural module (which, in this case, produces a cyclic expression pattern). The interface between these two modules is a single regulatory signal $(R_5$ in this case).

defined by the selective problem. I display the structure of the resulting gene networks using 'signaling' diagrams that includes both genes and regulatory products as nodes in the diagram. This emphasises the key role that the mediating regulatory products (transcription factors) play, providing an interface between upstream and downstream modules (see Figure 2).

I then show that these modules exhibit the following behaviour:

- Modules subject to a constant selective regime frequently undergo localised change via drift that affects how they compute the sub-task, but without changing the functional (input/output) profile of the module. This demonstrates a key feature of modularity: changes can be isolated with affecting the entire system.
- It is possible to independently select for modifications in either the upstream or downstream module without affecting the rest of the system. This further demonstrates the ability to isolate change within the system, this time under directional selection rather than drift.
- Lastly, I show that simulations that include multiple instances of these chained modules can be induced to coopt, or wire together, existing modules to produce new functions (see Figure 1 (d) and (e), and Figure 3). This demonstrates a second key function of modularity, its ability to recombine existing functional parts.

The structure of the chained task that these networks evolve to perform resembles the selective constraints under which multicellularity evolves, where each cell must integrate multiple upstream environmental cues to establish its

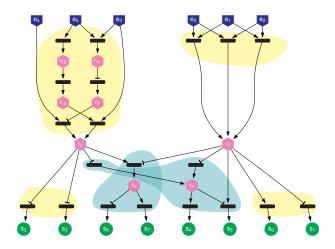


Figure 3: Upstream modules are first produced, along with coregulated structural genes (in yellow). A second selection regime then co-opts the output from these modules to use in new downstream structural modules (in blue).

position, and coordinate a variety of gene expression patterns on the basis of this position. Furthermore, the evolution of co-option in these networks closely resembles (indeed, it was inspired by) recent empirical work on the evolution of morphological novelties, where existing positional signals are co-opted to re-use existing pattern of downstream gene expression. The model results can thus provide some theoretical underpinning to recent attempts to formulate 'principles' of regulatory evolution derived only from empirical observation (Prud'homme et al., 2007).

Acknowledgements

This work was supported by the Australian Research Council, and Joshua Epstein's NIH Director's Pioneer Award, Number DP10D003874.

References

Calcott, B., Balcan, D., and Hohenlohe, P. a. (2008). A publishsubscribe model of genetic networks. *PloS one*, 3(9):e3245.

Clune, J., Mouret, J.-B., and Lipson, H. (2013). The evolutionary origins of modularity. *Proceedings of the Royal Society B: Biological Sciences*, 280(1755):20122863.

Kashtan, N. and Alon, U. (2005). Spontaneous evolution of modularity and network motifs. Proceedings of the National Academy of Sciences of the United States of America, 102(39):13773–8.

Prud'homme, B., Gompel, N., and Carroll, S. B. (2007). Emerging principles of regulatory evolution. *Proceedings of the National Academy of Sciences of the United States of America*, 104 Suppl:8605–12.

The Case for Engineering the Evolution of Robot Controllers

Fernando Silva^{1,3}, Miguel Duarte^{1,2}, Sancho Moura Oliveira^{1,2}, Luís Correia³ and Anders Lyhne Christensen^{1,2}

¹Instituto de Telecomunicações, Lisboa, Portugal
 ²Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal
 ³LabMAg, Faculdade de Ciências, Universidade de Lisboa, Portugal fsilva@di.fc.ul.pt

Abstract

In this paper, we discuss how the combination of evolutionary techniques and engineering-oriented approaches is an effective methodology for leveraging the potential of evolutionary robotics (ER) in the synthesis of behavioural control. We argue that such combination can eliminate the issues that have prevented ER from scaling to complex real-world tasks, namely: (i) the bootstrap problem, (ii) deception, (iii) the reality gap effect, and (iv) the prohibitive amount of time necessary to evolve controllers directly in real robotic hardware. We present recent studies carried out in our research group involving real-robot and simulation-based experiments. We provide examples of how the synergistic effects of evolution and engineering overcome each other's limitations and significantly extend their respective capabilities, thereby opening a new path in the design of robot controllers.

Introduction

Evolutionary computation techniques have been widely studied as a means to design robot controllers and body morphologies (Floreano and Keller, 2010), a field of research entitled evolutionary robotics (ER). ER has the potential to automate the synthesis of control systems. The experimenter relies on a self-organisation process, in which evaluation and optimisation of controllers is holistic, thereby avoiding the need for manual and detailed specification of the desired behaviour (Doncieux et al., 2011). The general idea is to optimise a population of genomes, each encoding a number of parameters of the robots' control system. Optimisation of genomes is based on Darwin's theory of evolution, namely blind variations and survival of the fittest, as embodied in the neo-Darwinian synthesis. The mapping from genotype to phenotype can capture different properties of the developmental process of natural organisms, and the phenotype can assume various degrees of biological realism (Stanley and Miikkulainen, 2003). Thus, ER draws inspiration from biological principles at multiple levels.

After approximately two decades of ER research, controllers have been evolved for robots with varied functionality, from terrestrial robots to flying robots (Floreano et al., 2005). Although there has been a significant amount of

progress in the field (Doncieux et al., 2011), it is arguably on a scale that has precluded ER techniques of being widely adopted. Evolved controllers are in most cases not yet competitive with human-designed solutions (Doncieux et al., 2011), and have only proven capable of solving relatively simple tasks such as obstacle avoidance, gait learning, and distinct searching tasks (Nelson et al., 2009). In effect, researchers have been consistently faced with a number of issues that must be addressed before ER becomes a viable approach, including: (i) the bootstrapping issues when solutions to more complex tasks are sought (Nelson et al., 2009), (ii) deception (Whitley, 1991), (iii) the reality gap (Jakobi, 1997), which occurs when controllers evolved in simulation become inefficient once transferred to the physical robot, and (iv) the prohibitively long time necessary to evolve controllers directly on real robots (Matarić and Cliff, 1996).

This paper is concerned with the synthesis of behavioural control for autonomous robots. We discuss the current limitations of ER, and we present directions for future research. We argue that it is conceivable to engineer the evolution of robotic controllers, i.e., to combine evolved solutions and human knowledge to better address the fundamental problems in ER. In effect, evolutionary algorithms are also engineered algorithms and, above all, the fitness function is usually the result of trial-and-error experiments involving a substantial amount of experimentation and human intervention. The key decision is therefore where to draw the line between human design and evolution. We argue that the role of evolution and of human expertise should be defined based on when the synthesis of controllers takes place, namely offline or online. We present recent research in our lab, and we show the synergistic effects and potential of this combined approach through a series of real robot and simulation-based experiments involving an e-puck robot (Mondada et al., 2009). By combining engineering-oriented approaches and evolutionary techniques, we successfully evolve controllers for three tasks: (i) a double T-maze rescue task, (ii) a tworoom cleaning task, and (iii) a deceptive phototaxis task. The main conclusion is that the proposed methodology is a viable new technique for leveraging the potential of ER.

Background and Related Work

In traditional ER approaches, controllers are synthesised *of-fline*, in simulation, to avoid the time-consuming nature of performing all evaluations on real robotic hardware. When a suitable controller is found, it is deployed on real robots. One of the central issues with the simulate-and-transfer approach is the reality gap (Jakobi, 1997), a frequent phenomenon in ER experiments. Controllers evolved in simulation can become inefficient once transferred onto the physical robot due to their exploitation of features of the simulated world that are different or that do not exist in the real world.

In *online evolution*, on the other hand, the evolutionary algorithm is executed on the robots themselves, while they perform their tasks. The main components of the evolutionary algorithm (evaluation, selection, and reproduction) are carried out autonomously by the robots without any external supervision. If the environmental conditions or task requirements change, the robots can modify their behaviour to cope with the new circumstances. However, the prohibitively long time required to evolve solutions on real robotic hardware is still a central impediment to large-scale adoption.

Besides the specific shortcomings of offline evolution and online evolution, there are two issues transversal to the two approaches: (i) the bootstrap problem (Gomez and Miikkulainen, 1997), and (ii) deception (Whitley, 1991). Bootstrapping issues occur when the task is too demanding to apply any meaningful selection pressure on a randomly generated population of candidate solutions. All individuals in the early stages of evolution may perform equally poorly, and evolution drifts in an uninteresting region of the search space. Deception occurs when the fitness function fails to build a gradient that leads to a global optimum, and instead drives evolution towards local optima. The more complex the task, the more susceptible is evolution to deception (Lehman and Stanley, 2011). Consequently to all these issues, ER techniques do not yet scale to tasks with the level of complexity found outside strictly controlled laboratory conditions (Nelson et al., 2009). The next sections review the current approaches introduced in ER for dealing with the problems discussed above.

Crossing the Reality Gap

Miglino et al. (1996) proposed three complementary approaches to cross the reality gap: (i) using samples from the real robots' sensors to enable more accurate simulations, (ii) introducing a conservative form of noise in simulated sensors and actuators to reduce the performance gap between the simulated and the real world, and (iii) continuing evolution for a small amount of time in real hardware if a decrease in performance is observed when controllers are transferred. The sensor sampling and the conservative noise methods have since become widespread. Continuing evolution in real hardware has not been frequently used, despite pioneering work in this direction (Nolfi et al., 1994).

Jakobi (1997) advocated the use of *minimal simulations*, in which the experimenter only implements features of the real world deemed needed for successful evolution of controllers. The remaining features are hidden in an "envelope of noise" to minimise the effects of simulation-only artifacts. It is not clear if Jakobi's approach scales well to complex tasks, since such tasks: (i) naturally involve more robotenvironment interactions, and therefore more features, and (ii) require that the experimenter can determine the set of relevant features and build a task-specific simulation model.

Recently, Koos et al. (2013) introduced the *transferability approach*, in which controllers are evaluated based on their combined simulation and real-robot performance. To avoid testing each candidate solution in a real robot, a surrogate model is created and then updated periodically based on the results of real-robot experiments. The transferability approach has been shown to work when a solution can be found in relatively few generations (100 or less), but it can become unfeasible once the task requires several hundreds or thousands of generations with long evaluations. Furthermore, the difficulties in automatically evaluating controllers in real hardware represent an additional challenge.

Overcoming the Bootstrap Problem and Deception

Over the years, different approaches have been proposed to solve increasingly more complex tasks. In incremental evolution, the experimenter decomposes a task to bootstrap evolution and circumvent deception. There are numerous ways to apply incremental evolution (Mouret and Doncieux, 2008), such as dividing the task into sub-tasks that are solved sequentially, or making the task progressively more difficult through environmental complexification (Christensen and Dorigo, 2006). Although incremental evolution can be seen as an approach in which engineering and evolution are combined, it is typically performed in an unstructured manner. The experimenter has to perform a manual switch between the execution of each component of the evolutionary setup, such as different sub-tasks, which can significantly affect the global performance of solutions evolved (Mouret and Doncieux, 2008). In addition, if the components of the setup are highly integrated, incremental evolution can be difficult to apply successfully (Christensen and Dorigo, 2006).

Lehman and Stanley (2011) introduced *novelty search*, in which the idea is to maximise the novelty of behaviours instead of their fitness, i.e., to search directly for novel behaviours as a means to circumvent convergence to local optima. A number of studies outlined that novelty search is unaffected by deception, less prone to bootstrapping issues, and can evolve simpler solutions than those evolved by traditional fitness-based optimisation (Lehman and Stanley, 2011). Novelty search is, however, significantly dependent on the *behaviour characterisation* (Kistemaker and Whiteson, 2011), and can be challenging to apply when such a metric is not easy to define. That is, although novelty search

operates independently of fitness, its effectiveness is dependent on a similar form of human knowledge, despite recent studies involving generic characterisations (Gomes and Christensen, 2013).

Recently, a human-in-the-loop approach for avoiding deception was introduced by Celis et al. (2013). The approach allows non-expert users to guide evolution away from local optima by indicating intermediate states that the robot must go through during the task. A gradient is then created to guide evolution through the states. This approach was demonstrated in a deceptive object homing task, and it is still unknown if it generalises to different types of tasks.

Evolution in Physical Hardware

The first example of online evolution in a real, neural network-driven mobile robot was performed by Floreano and Mondada (1996). The authors successfully evolved navigation and homing behaviours for a Khepera robot. The studies were a significant breakthrough as they showed the possibility of online evolution of robot behaviour. Researchers then focused on the challenges posed by evolving controllers directly on physical robots, with a special focus on the prohibitively long time required (Matarić and Cliff, 1996). Afterwards, Watson et al. (2002) introduced *embodied evolution*, in which the use of multirobot systems was motivated by an anticipated speed-up of evolution due to the inherent parallelism in such systems.

Over the past decade, different approaches to online evolution have been proposed (Silva et al., 2012). Notwithstanding, few studies have been conducted on real robots. Researchers have focused on developing different evolutionary approaches and evaluating them mainly through online evolution in simulation. Despite the algorithmic advances, the strikingly long time that the online evolutionary process still requires during complex experiments renders the approach infeasible.

Engineering the Evolution of Controllers

The main objective of our ongoing work is to enable ER techniques to scale to more complex tasks by minimising the current issues in the field. We propose the *systematic* use of more practical, engineering-oriented approaches in which the significant potential of evolution in controller design is leveraged by human knowledge.

An engineering methodology in ER has not yet been agreed upon (Trianni and Nolfi, 2011). For instance, while different studies have combined evolved control and preprogrammed control, it is usually done in an ad-hoc manner, see Groß et al. (2006) for example, or by imposing hard behaviour-based architectures in which the role of evolution is minimal, see Urzelai et al. (1998). Contrary to such approaches, we argue that there is a *context-dependent compromise* between engineering and evolution. When conducting evolution offline, the experimenter has complete control

over the experimental conditions and can modify and correct the selection pressures. Furthermore, the experimenter can take a methodical approach to find a suitable fitness function, an appropriate controller structure, or explore different evolutionary algorithms. That is, evolution is put at the service of engineering.

Complementarily, when evolving controllers online, the evolutionary algorithms run autonomously from the start and execute without any kind of human supervision. However, the experimenter can seed evolution with a bias towards certain types of solutions or behaviours, thereby inserting specific human knowledge into the evolutionary search. That is, the experimenter can give evolution *direct access* to task-related competences that are engineered before online evolution is conducted. If the structure and the parameters of these competences are under evolutionary control, they can be optimised during task execution, and evolution can progressively complexify controllers by using these building blocks as a substrate. In this way, engineering is put at the service of evolution.

At first sight, one may argue that the above perspectives imply an antagonistic relationship between offline evolution and online evolution. However, depending on the task complexity and requirements, offline evolution and online evolution may complement each other. In relatively simple tasks, it may be indifferent to conduct evolution offline or online. As the complexity of the task increases, the issues of each approach are exacerhated: (i) the more complex the controller, the more difficult it is to ensure successful transfer from simulation to reality, and the more time-consuming is evolution directly on real hardware, and (ii) in both cases, the more prone is evolution to bootstrap issues and deception. One solution is to exploit the benefits of each approach to bypass each other's limitations. Offline evolution can be used as an initialisation procedure in which approximate, yet effective solutions are engineered and deployed to real robots. During task execution, online evolution can serve as a refinement procedure that enables robots to adapt to changing or unforeseen circumstances.

In the following sections, we describe two complementary approaches for engineering ER: the hierarchical controller approach for offline evolution and the macro-neurons approach for online evolution, and we discuss how our approaches can mitigate the current issues in ER.

Engineering Offline Evolution

The hierarchical controller approach relies on a systematic hierarchical decomposition of the task, and structured composition of controllers that can be either evolved or preprogrammed (Duarte et al., 2014). We divide the task into simpler sub-tasks when evolution is unable to find a solution to a given task. Sub-controllers are evolved or preprogrammed to solve each sub-task, and the complete controller is composed in a hierarchical, bottom-up manner as shown

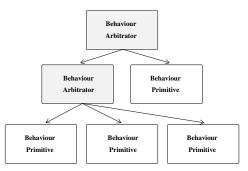


Figure 1: Representation of a hierarchical controller. Behaviour arbitrators determine which sub-controller to execute, and behaviour primitives control the actuators.

in Fig. 1. Each node in the hierarchy is either a *behaviour* arbitrator or a *behaviour* primitive (Lee, 1999). Behaviour primitives are at the bottom of the controller hierarchy and control a number of actuators of the robot, while behaviour arbitrators determine which primitive to execute at a given time. The logic in each node is independent of the logic in other nodes. Thus, evolved nodes can be synthesised by different evolutionary processes.

The evolution of behaviour primitives is based on the concept of an appropriate fitness function, which: (i) enables evolution to bootstrap, (ii) leads to controllers that consistently and efficiently solve the task in simulation, and (iii) evolves controllers that are able to maintain their performance levels in real robotic hardware. Provided an appropriate fitness function can be defined for a given task, we evolve a behaviour primitive composed of a single ANN. Otherwise, we recursively divide the task into sub-tasks until appropriate fitness functions have been found for each sub-task. Behavioural primitives are manually programmed when: (i) a sub-task cannot be further divided and an appropriate fitness function cannot be found, or (ii) if a particular robot-environment interaction is too difficult to accurately simulate. After the synthesis of behaviour primitives, sub-controllers are created by evolving or programming behaviour arbitrators in a bottom-up fashion. Each behaviour arbitrator receives a number of sensory inputs and is responsible for delegating control to the level below. Subcontrollers are then combined with other sub-controllers until the hierarchical controller is completed. Each time a new sub-controller has been synthesised, its performance on real robotic hardware can be evaluated, which allows to address transfer-related issues in an incrementally manner during the development of the control system.

An important aspect of our approach is that, as we move up the controller hierarchy and attempt to synthesise controllers for increasingly complex tasks, appropriate fitness functions may be increasingly difficult to define. In such cases, the fitness function can be *derived* based on the task decomposition and constructed to reward the arbitrator for activating a valid sub-controller for the current sub-task,

rather than for solving the complete task. Thus, while previous studies have hierarchically decomposed controllers based on different techniques, from genetic programming to neuroevolution, see Duarte et al. (2014) for a review, our approach is distinct in a number of aspects. Firstly, we synthesise hybrid controllers in which preprogrammed control and evolved control can be seamlessly integrated, thus compounding the benefits of ER in the design of controllers, and preprogrammed behaviours that would otherwise be difficult or infeasible to evolve. Secondly, we use derived fitness functions to circumvent the otherwise increase in fitness function complexity. Finally, we bypass bootstrapping and deception-related issues due to the hierarchical task decomposition.

Engineering Online Evolution

This section introduces the macro-neurons approach for online evolution of neural network-based controllers (Silva et al., 2014). In this approach, neural networks use standard neurons as elementary components, and higher level units representing behaviours called the macro-neurons.

Each macro-neuron M is defined by (I,O,f,P), where I and O are respectively the set of input connections and of output connections, f is the function computed by the macro-neuron, and P is the set of parameters that can be optimised through evolution. Each connection $I_i \in I$ contains a weight $w_i \in w$ and transmits to M an input value $x_i \in x$. The computation of M is given by f(w,x) = y, where y is the output vector of M, and each $y_j \in y$ is transmitted to other neurons via the corresponding connection $O_j \in O$. Depending on the type of the macro-neuron M, the set P contains different elements. If M is an evolved ANN, then P refers to the connections and neurons that can be modified by evolution; if M is preprogrammed, P contains the parameters of the behaviour, if any.

In our approach, the macro-neurons are prespecified in the neural architecture before online evolution is conducted. The construction of ANNs using macro-neurons is shown in Fig. 2. Figure 2a illustrates how different preprogrammed macro-neurons are specified. Each macro-neuron transmits two values to each output neuron: (i) an activity value representing the signal to be sent to the actuators controlled by the output neurons, and (ii) a priority value, which represents the effective need of the behaviour to execute at a given time. Priority and activity values are used to better resolve conflicts when different preprogrammed macroneurons compete for control (Silva et al., 2014). Complementarily, Fig. 2b shows how an evolved ANN is represented as a macro-neuron. The connections from the macroneuron to the output neurons enable evolution to arbitrate and shape the output values of different macro-neurons.

In the experiments described in the following section, the macro-neurons are used in combination with odNEAT (Silva et al., 2012), an online neuroevolutionary algorithm that

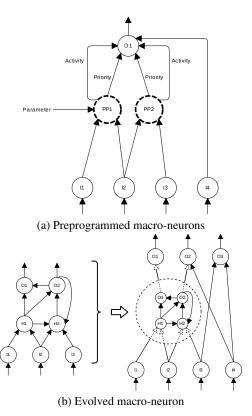
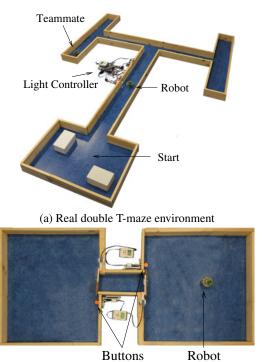


Figure 2: Examples of the integration of different types of macro-neurons in neural architectures. (a) Two preprogrammed macro-neurons. (b) An evolved ANN-based macro-neuron inserted into a larger controller.

evolves the weights and the topology of ANNs in single and multirobot systems (Silva et al., 2012, 2014). Thus, the evolutionary process can: (i) adapt the preprogrammed behaviours by adjusting their parameters, see PP1 in Fig. 2a, (ii) modulate the execution of macro-neurons by increasing or decreasing the strength of connections, including those related to the priority and activity values, and (iii) optimise evolved ANN-based macro-neurons and the entire network by augmenting their structure and by adjusting the connection weights. By combining ANNs and macro-neurons, we compound: (i) the ANNs' robustness and tolerance to noise, (ii) the benefits of each type of macro-neuron, which can be synthesised by distinct evolutionary processes or manually designed to shortcut complex evolutionary processes, (iii) higher level bootstrapping, which can enable robots to adapt to complex and dynamic tasks in a timely manner.

Experimental Results and Discussion

In this section, we assess the viability of our approaches in both real-robot and simulation-based single robot experiments. In our experiments, we use an e-puck (Mondada et al., 2009), a 7.5 cm in diameter differential drive robot capable of moving at a maximum speed of 13 cm/s. The experiments were introduced in Duarte et al. (2012, 2014) and



(b) Real two-room cleaning environment

Figure 3: The environments in which the hierarchical controller is assessed: (a) double T-maze with size of 2 x 2 meters, and (b) the two-room cleaning environment. The rooms are connected by a corridor blocked by two doors. Each room has one button that can be pushed to open the doors.

Silva et al. (2014). We review our previous results, and we argue the importance and effectiveness of combining engineering and evolution in the synthesis of robotic controllers.

Offline Design of Hierarchical Controllers

In this section, we apply the hierarchical controller approach to solve two tasks: (i) a rescue task in a double T-maze (Duarte et al., 2012), and (ii) a dust cleaning task (Duarte et al., 2014). The two environments are shown in Fig. 3. In the rescue task, the robot must exit a room with a number of obstacles, solve the T-maze, find the teammate, and safely guide the teammate back to the initial room. Two rows of flashing lights in the main corridor of the double T-maze give the robot information by indicating the branch leading to the teammate. In the dust cleaning task, dust spots appear in two rooms that are connected by a corridor. A new dust spot is randomly placed in one of the rooms every 10s, up to a maximum of five dust spots in the environment at any given time. Each room has one button that can be pushed to open the doors that give access to the corridor.

We first tried to evolve a monolithic controller for the complete rescue task using: (i) a standard $(\mu + \lambda)$ evolution strategy that optimised the weights of fixed-topology continuous-time recurrent neural networks with one hidden layer of fully-connected neurons, and (ii) the prominent

NEAT algorithm (Stanley and Miikkulainen, 2002), which evolves both the neural network's weights and topology. While the controllers evolved by the respective algorithms successfully solved initial parts of the task, none of them was able to complete the entire rescue task in simulation. We therefore divided the rescue task into three sub-tasks: (i) exit the room, (ii) navigate through the double T-maze and find the teammate, and (iii) return to the initial room while guiding the teammate. We decomposed the control system into three main sub-controllers: "Exit Room" primitive, "Solve Maze" arbitrator, and "Return to Room" arbitrator. Both the "Solve Maze" and the "Return to Room" arbitrators had access to three locomotion behaviour primitives: "Follow Wall", "Turn Left" and "Turn Right". A top-level arbitrator was evolved to select which sub-controller to activate at any given time. The controllers achieved an average success rate of 85%. The highest scoring hierarchical controller solved the task 93% of the times in simulation and of 92% in real robotic hardware.

To solve the two-room cleaning task, we decomposed the control system into two main sub-controllers: an evolved "Change Room" arbitrator and an evolved "Clean" primitive. The "Change Room" arbitrator was given access to an evolved "Open Door" arbitrator and to an evolved "Enter Corridor" primitive. The "Open Door" arbitrator had access to an evolved "Go To Button" primitive and to a preprogrammed "Push Button" primitive. Thus, all arbitrators and primitives were evolved, except for the "Push Button" primitive. Pushing a button to open the doors requires fine sensorimotor coordination, since the buttons are difficult to detect and hit. As this is a difficult interaction to model correctly in simulation, and therefore a behaviour to evolve and transfer successfully, the "Push Button" primitive was preprogrammed. In the complete task, the hierarchical controllers were evaluated according to the number of dust spots they cleaned in five minutes of real and simulated time. The controllers displayed high performance levels as they cleaned an average of 18.74 dust spots in simulation, and an average of 18.44 dust spots on the real e-puck robot.

We successfully synthesised controllers to solve two tasks with different requirements. One of the main ideas behind our approach is that ER techniques should not be applied blindly. We proposed an engineering methodology that exploits the knowledge acquired from negative results when a suitable controller cannot be evolved, and enables the decomposition of the task into simpler sub-tasks on an asneeded basis. By taking a systematic approach that combines evolution with engineering, we were able to overcome three fundamental issues: (i) the bootstrap problem, (ii) deception, and (iii) the reality gap, as the controllers maintained their performance levels in real robotic hardware. The bootstrapping problem and deception are naturally bypassed by dividing a complex task into simpler sub-tasks. The success in crossing the reality gap is due to the hand-design of

sub-controllers when necessary and to the iterative tests of evolved sub-controllers (Duarte et al., 2014), in which the experimenter can address transfer-related issues locally in the controller hierarchy.

Additionally, it should be noted that by recursively focusing on controllers for simpler sub-tasks, the experimenter can more easily encourage the evolution of robust solutions that operate effectively in a large number of environmental conditions and that maintain their performance levels on real robots. In this way, solutions evolved can be made more general and therefore better sustain conditions not seen during evolution. As a final remark, it is worth discussing the role of human knowledge in our approach. In standard ER experiments, evolutionary setups are often found in an adhoc manner. The experimenter has to determine a suitable fitness function, the controller type and structure, the evolutionary algorithm, and the parameters associated with the evolutionary algorithm through a trial-and-error process. All these components are hand-designed, and usually involve a substantial amount of experimentation and human intervention. Contrary to unregulated trial-and-error methods, we follow a structured approach in which human knowledge is used to actively eliminate the factors that limit evolution and guide it towards classes of controllers relevant to the task.

Online Evolution with Macro-neurons

To assess the macro-neurons approach, we study a single robot deceptive and dynamic version of the phototaxis task with three light sources (Silva et al., 2014). The task environment is shown in Fig. 4. The robot has a constant virtual energy consumption value. The light sources are sensed by the robot within a 25 cm range. One source is beneficial to the robot as it increases the energy level, one source is neutral, and the remaining source is detrimental as it decreases the energy level. The sources are static, but they switch their type in a clockwise manner at five minute intervals. Deceptiveness is introduced by the fact that the three light sources are indistinguishable to the robot's light sensors. Thus, the robot must discriminate between the different sources based on the temporal correlation between its energy sensor readings and proximity to a given source.

We conducted experiments using two types of macroneurons: evolved ANNs, synthesised offline using NEAT (Stanley and Miikkulainen, 2002), and preprogrammed behaviours. We synthesised three basic primitives of each type: (i) a move forward behaviour, (ii) a turn left behaviour, and (iii) a turn right behaviour. We conducted four sets of experiments: (i) evolution without macro-neurons, (ii) and (iii) evolution with access respectively to the preprogrammed and the evolved macro-neurons, and (iv) an hybrid approach involving a preprogrammed "Move Forward" macro-neuron and two evolved "Turn" behaviours.

The experimental setups involving macro-neurons enabled an efficient synthesis of controllers, with the advan-

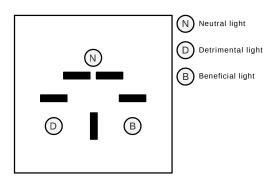


Figure 4: The task environment. The arena measures 3 x 3 meters. The dark areas denote obstacles, while the circular areas represent the different sources. The distance between the sources is set at 1.5 meters.

tage being in favour of evolved macro-neurons. Given the deceptiveness and complexity of the task, evolution without access to macro-neurons required an average of 9.50 hours of simulated time to evolve controllers that solve the task. The three types of controllers with macro-neurons required between 1.78 hours and 2.91 hours, thereby reducing the evolution time between 53% and 80%. In addition, the use of macro-neurons yielded competitive or superior solutions in terms of the fitness score (Silva et al., 2014).

Our current results suggest that approaches such as the macro-neurons may be a viable solution to speed-up online evolution in real robotic hardware. The key idea is that the experimenter can compensate for the absence of control he or she has during online evolution experiments by biasing evolution towards desired classes of behaviour. In effect, macro-neurons need not only to represent task-oriented behaviours. The macro-neurons can also represent prespecified survival-oriented behaviours that enable, for instance: (i) a group of robots to coordinate and share the access to battery charging stations, a task that been found to be highly deceptive (Gomes and Christensen, 2013), and (ii) to selfpreserve by minimising collisions and hardware damage. By giving evolution access to these fundamental building blocks of distinct complexity and with different functions, bootstrapping is made easier because partial solutions to the task are already available. Additionally, evolution can focus on combining the engineered building blocks with evolved behaviours to synthesise increasingly sophisticated action patterns. New competences can be integrated in a scalable manner by gradually expanding the behavioural repertoire of the robot. Thus, rather than attempting to develop a purely automatic and potentially less efficient online evolutionary algorithm, the experimenter can take advantage of his or her knowledge to determine what are the basic components to solve the task. Each of the components can then be used by evolution in the search for a complete controller.

Intuitively, seeding evolution with specific behavioural properties may restrict the search space, and therefore potentially represents a trade-off between the adaptation time and the generality of behaviours that can be evolved. Nonetheless, recent experiments (Silva et al., 2014) have shown that evolution may be able to successfully adapt and reuse macro-neurons that are less optimised or even unsuited to the task. Additional experiments with different tasks are required to successfully answer this question.

Conclusions and Future Work

In this paper, we have argued that the combination of engineering-oriented and evolutionary approaches can minimise the current issues in ER, namely: (i) the bootstrap problem, (ii) deception, (iii) the reality gap, and (iv) the long time required for online evolution experiments. There are multiple reasons why our proposed methodology represents a valuable design tool, one of the most important being that the experimenter can influence how human knowledge and evolution are combined. In this way, the advantages of engineering-oriented and evolutionary approaches can be united to more easily overcome each other's limitations.

We presented two methods that combine the strengths of evolution and engineering: (i) the hierarchical controller approach, and (ii) the macro-neurons approach. The incorporation of evolution and engineering resulted in an effective synergy that enabled us to successfully evolve controllers for three tasks with a number of different traits. An important methodological advantage of our approaches is that they can be combined if deemed necessary. Hierarchical controllers of distinct complexity and functionality can also be encapsulated in a macro-neuron and adapted online. This versatility moves engineering and evolution from the space of offline or online synthesis of controllers to the space of offline approximation and online refinement of solutions. Thus, the key contribution of this paper is that our methodology is a flexible and viable approach for scaling evolutionary robotics to more complex tasks, without burdening the experimenter with the responsibility of performing a manual and detailed specification of the desired behaviour.

We are currently assessing our methodology in a varied set of tasks that have proven challenging for existing techniques, such as those that require a fine sensorimotor coordination. Because in more complex tasks, the division of a task into sub-tasks may not be intuitive, we are working towards having the evolutionary algorithm to perform the task decomposition itself. We are also extending our methods to multirobot systems to take advantage of the properties of decentralisation and robustness that pertain to self-organising systems. The main objective of our research is to reduce the current gap between ER and "mainstream" robotics.

Acknowledgements This work was partially supported by the Fundação para a Ciência e Tecnologia (FCT) under the grants SFRH/BD/76438/2011, SFRH/BD/89573/2012, PEst-OE/EEI/LA0008/2013, PEst-OE/EEI/UI0434/2014, and EXPL/EEI-AUT/0329/2013.

References

- Celis, S., Hornby, G. S., and Bongard, J. (2013). Avoiding local optima with user demonstrations and low-level control. In Proceedings of the IEEE Congress on Evolutionary Computation, pages 3403–3410. IEEE Press, Piscataway, NJ.
- Christensen, A. L. and Dorigo, M. (2006). Incremental evolution of robot controllers for a highly integrated task. In *Proceedings of the Ninth International Conference on the Simulation* of Adaptive Behavior, pages 473–484. Springer, Berlin, Germany.
- Doncieux, S., Mouret, J.-B., Bredeche, N., and Padois, V. (2011). Evolutionary robotics: Exploring New Horizons, volume 341 of Studies in Computational Intelligence, chapter 1, pages 3–25. Springer, Berlin, Germany.
- Duarte, M., Oliveira, S., and Christensen, A. L. (2012). Hierarchical evolution of robotic controllers for complex tasks. In *Proceedings of the IEEE International Conference on Development and Learning and on Epigenetic Robotics*, pages 1–6. IEEE Press, Piscataway, NJ.
- Duarte, M., Oliveira, S. M., and Christensen, A. L. (2014). Evolution of hybrid robotic controllers for complex tasks. *Journal of Intelligent and Robotic Systems, in press*.
- Floreano, D. and Keller, L. (2010). Evolution of adaptive behaviour by means of Darwinian selection. *PLoS Biology*, 8(1):e1000292.
- Floreano, D. and Mondada, F. (1996). Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics B*, 26(3):396–407.
- Floreano, D., Zufferey, J.-C., and Nicoud, J.-D. (2005). From wheels to wings with evolutionary spiking circuits. *Artificial Life*, 11(1-2):121–138.
- Gomes, J. and Christensen, A. L. (2013). Generic behaviour similarity measures for evolutionary swarm robotics. In *Proceedings of the Fifteenth Genetic and Evolutionary Computation Conference*, pages 199–206. ACM Press, New York, NY.
- Gomez, F. and Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior*, 3-4:317–342.
- Groß, R., Bonani, M., Mondada, F., and Dorigo, M. (2006). Autonomous self-assembly in swarm-bots. *IEEE Transactions on Robotics*, 22(6):1115–1130.
- Jakobi, N. (1997). Evolutionary robotics and the radical envelopeof-noise hypothesis. *Adaptive Behavior*, 6(2):325–368.
- Kistemaker, S. and Whiteson, S. (2011). Critical factors in the performance of novelty search. In *Proceedings of the Thirteenth Genetic and Evolutionary Computation Conference*, pages 965–972. ACM Press, New York, NY.
- Koos, S., Mouret, J.-B., and Doncieux, S. (2013). The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Transactions on Evolutionary Computation*, 17(1):122–145.
- Lee, W.-P. (1999). Evolving complex robot behaviors. *Information Sciences*, 121(1-2):1–25.

- Lehman, J. and Stanley, K. O. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223.
- Matarić, M. and Cliff, D. (1996). Challenges in evolving controllers for physical robots. *Robotics and Autonomous Systems*, 19(1):67–83.
- Miglino, O., Lund, H. H., and Nolfi, S. (1996). Evolving mobile robots in simulated and real environments. *Artificial Life*, 2(4):417–434.
- Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C., Floreano, D., and Martinoli, A. (2009). The e-puck, a robot designed for education in engineering. In *Proceedings of the Ninth Conference on Autonomous Robot Systems and Competitions*, pages 59–65. IPCB, Castelo Branco, Portugal.
- Mouret, J.-B. and Doncieux, S. (2008). Incremental evolution of animats' behaviors as a multi-objective optimization. In *Proceedings of the Tenth International Conference on the Simulation of Adaptive Behaviour*, pages 210–219. Springer, Berlin, Germany.
- Nelson, A., Barlow, G., and Doitsidis, L. (2009). Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57(4):345–370.
- Nolfi, S., Floreano, D., Miglino, O., and Mondada, F. (1994). How to evolve autonomous robots: Different approaches in evolutionary robotics. In *Proceedings of the Fourth International Workshop on the Synthesis & Simulation of Living Systems*, pages 190–197. MIT Press, Cambridge, MA.
- Silva, F., Correia, L., and Christensen, A. L. (2014). Speeding up online evolution of robotic controllers with macro-neurons. In *Proceedings of the Sixteenth European Conference on the Applications of Evolutionary Computation*. Springer, Berlin, Germany. In press.
- Silva, F., Urbano, P., Oliveira, S., and Christensen, A. L. (2012). odNEAT: An algorithm for distributed online, onboard evolution of robot behaviours. In *Proceedings of the Thirteenth International Conference on the Simulation & Synthesis of Living Systems*, pages 251–258. MIT Press, Cambridge, MA.
- Stanley, K. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.
- Stanley, K. and Miikkulainen, R. (2003). A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130.
- Trianni, V. and Nolfi, S. (2011). Engineering the evolution of self-organizing behaviors in swarm robotics: A case study. *Artificial Life*, 17(3):183–202.
- Urzelai, J., Floreano, D., Dorigo, M., and Colombetti, M. (1998). Incremental robot shaping. *Connection Science*, 10(3-4):341–360.
- Watson, R., Ficici, S., and Pollack, J. (2002). Embodied evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, 39(1):1–18.
- Whitley, L. (1991). Fundamental principles of deception in genetic search. In *First Workshop on Foundations of Genetic Algorithms*, pages 221–241. Morgan Kaufmann, San Mateo, CA.

Self-Replicating Distributed Virtual Machines

Lance R. Williams¹

¹Department of Computer Science, University of New Mexico, Albuquerque, NM 87131 williams@cs.unm.edu

Abstract

Recent work showed how an expression in a functional programming language can be compiled into a massively redundant asynchronous spatial computation called a distributed virtual machine. A DVM is comprised of bytecodes reified as actors undergoing diffusion and communicating via messages containing encapsulated virtual machine states. Significantly, it was shown that both the efficiency and the robustness of expression evaluation by DVM increase with redundancy. In the present work, spatial computations that become more efficient and robust over time are described. They accomplish this by self-replication, which increases the redundancy of the elements of which they are comprised. The first and simplest of these self-replicating DVMs copies itself by reflection; it reads itself from a contiguous range of memory. The remainder are quines. As such, they reproduce by translating and transcribing self-descriptions. The nature of the self-descriptions and of the translation and transcription processes differ in each case. The most complex selfreplicating DVM described represents a fundamentally new kind of artificial organism-a machine language program reified as a spatial computation that reproduces by compiling its own source-code.

Introduction

In a recent article, Ackley (2013) argues that future robust computing systems will be defined in terms of *bespoke physics interfaces* built on top of a physical substrate resembling an *asynchronous cellular automaton (ACA)*; space and time in *indefinitely scalable* computations will be *coextensive* with physical space and time. von Neumann, who invented the *random access stored program computer (RASP)*, showed us that programs can be data. Yet conventionally, machines are active and data is passive; machines *transform* data. In this paper, we propose that computations formulated in terms of a bespoke physics are *literally* machines and demonstrate how self-replicating programs written in a Lisp-like language can be compiled into such computations.

A range of computational substrates have been used to host self-replicating programs in artificial life research or might play this role in the future. The *distributed virtual machine (DVM)* developed by the author as a method for the robust evaluation of expressions is in the second category

Table 1: Computational substrates for artificial life.

	control	sites	bits/site	r/w dist.
CA	C	$N \times N$	O(1)	O(1)
ACA	D	$N \times N$	O(1)	O(1)
DVM	D	$N \times N \ge M$	$O(\log M)$	O(1)
Avida	C	$N \times N$	$O(P) + Q \log P$	P
Tierra	C	P	$O(1) + Q \log P$	P
RASP	C	1	$M \log M$	М

(Williams, 2012). See Table 1.

Notwithstanding their historical importance, CAs are likely to be supplanted by ACAs for two reasons. First, their dependence on a global clock violates Ackley's requirement that a bespoke physics be indefinitely scalable. Second, it is well known that for every CA there exists an ACA which emulates it (Nakamura, 1974; Nehaniv, 2004). Less well known is the fact that emulations of this type can be done with negligible slowdown (Berman and Simon, 1988).

Tierra (Ray, 1994) and Avida (Adami et al., 1994) exemplify the artificial life approach to evolutionary computation. Both systems are based on a *virtual machine (VM)* that has two distinct address spaces. The first (used to hold programs) consists of P words of size O(1) bits while the second (used to hold a stack of program memory addresses at runtime) consists of Q words of size $\log P$ bits. In contrast, the DVM and RASP both have a single address space consisting of M words of size $\log M$ bits; the major difference between the DVM and RASP is that the DVM's memory is spatially distributed over an $N \times N$ grid.

Moreover, like the RASP, the Tierra and Avida VMs are random access, with instructions that can read and write values anywhere in a memory of size P. In contrast, a DVM can be implemented on an ACA substrate with only $O(\log M)$ bits per site and with a read/write distance which is O(1).

Finally, the table reveals an important difference between Tierra and Avida, namely that all organisms occupy a single address space in Tierra but are segregated in separate address spaces in Avida. The difference is not minor—in Tierra the competition for program memory *P* is zero sum while in Avida it isn't. However, organisms which evolve more effi-

cient methods of self-replication in terms of stack memory Q go unrewarded in both systems. In contrast, in a DVM, the competition for heap-allocated space M (no matter its use) among all organisms sharing the $N \times N$ grid is zero sum.

A Simple Programming Language

The language we used to construct our self-replicating DVMs is a pure functional subset of Scheme which we call *Skeme*. Because it is purely functional, *define*, which associates values with names in a global environment using mutation, and *letrec*, which also uses mutation, have been excluded. The global environment itself is eliminated by making primitive functions constants. For simplicity, closures are restricted to one argument; user defined functions with more than one argument must be written in a curried style. This simplifies the representation of the lexical environment which is used at runtime by making all variable references integer offsets into a flat environment stack; these are termed *de Bruijn indices* and can be used instead of symbols to represent bound variables (De Bruijn, 1972).

One feature peculiar to Skeme is the special-form, *lambda+*. When a closure is created by *lambda+*, the closure's address is added to the front of the enclosed environment; the de Bruijn index for this address can then be used for recursive function calls. For example, the following function computes factorial:

```
(lambda+ (if (= %0 0) 1 (* %0 (%1 (- %0 1)))))
```

where %0 is a reference to the closure's argument and %1 is a reference to the closure's address.

Evaluation of Expressions by Virtual Machines

The process of evaluating expressions by compiling them into bytecodes which are executed on a VM was first described by Landin (1964) for Lisp and was generalized for Scheme by Dybvig (1987). Because it plays an important role in our work, it is worth examining Dybvig's model for Scheme evaluation in some detail.

Evaluating an expression requires saving the current evaluation context onto a stack, then recursively evaluating subexpressions and pushing the resulting values onto a second stack. The second stack is then reduced by applying either a primitive function or a closure to the values it contains. Afterwards, the first stack is popped, restoring the prior evaluation context. Expressions are compiled into trees of bytecodes which perform these operations when the bytecodes are interpreted. For book keeping during this process, Dybvig's VM requires five registers. See Figure 1.

With the exception of the *accumulator*, which can point to an expression of any type, and the *program counter*, which points to a position in the tree of bytecodes, each of the registers in the VM points to a heap allocated data structure comprised of pairs; the *environment* register points to a stack representing definitions in enclosing lexical scopes, the *arguments* register points to the stack of values which a func-

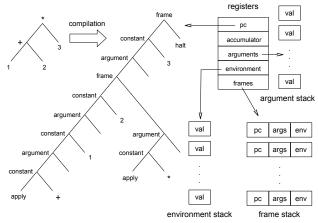


Figure 1: Virtual machine for evaluating compiled Skeme expressions showing its registers and associated heap-allocated data structures (based on Dybvig (1987)).

tion (or closure) is applied to, and the *frames* register points to a stack of suspended evaluation contexts.

Evaluation occurs as the contents of these registers are transformed by the interpretation of the bytecodes. For example, the *argument* bytecode pushes the value of an evaluated subexpression onto the argument stack. Other bytecodes alter the frame stack. For example, the *frame* bytecode pushes an evaluation context onto the frame stack, while the *apply* bytecodes pops it after applying a primitive function (or a closure) to the values found in the argument stack. Still other bytecodes load the accumulator. For example, the *constant* bytecode loads it with a constant, while the *refer* bytecode loads it with a value found in the environment stack.

Reified Actor Models

Actors are universal primitives for constructing concurrent computations introduced by Hewitt et al. (1973). In essence, an actor is a lightweight process with a unique address which can send and receive messages to and from other actors. In response to receiving a message, and (depending on the message's contents) an actor can send a finite number of messages of its own; create a finite number of new actors; and change its internal state so that its future behavior is different. All of these things happen asynchronously.

Although as originally conceived, actor models are not reified, it is possible to create a *reified actor model*. In a reified actor model, all actors have unique positions on a 2D grid. Actors possess a finite number of states and can sense and change the positions and states of actors in their $n \times n$ neighborhoods. Significantly, actors can create bonds with other actors; bonds are *relative spatial addresses* which are short, symmetric, and automatically updated as actors undergo random independent motion or *diffusion*. Since bonds are short, they restrict the motion of an actor which possesses them. However, they also ensure that two actors which must communicate can always do so.

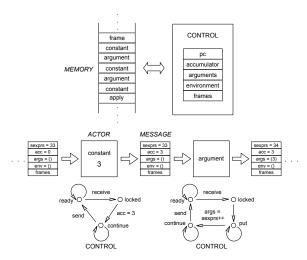


Figure 2: Conventional *virtual machine* (top) and *distributed virtual machine* (bottom). In the DVM, the registers are encapsulated in a message called a *continuation* that is passed between bytecodes reified as actors. Each actor is a finite state machine that transforms the continuation in manner specific to its type then passes it to the next bytecode in the program. Control is distributed not centralized.

Distributed Virtual Machines

By giving them addresses and types, reified actors can be used to represent heap-allocated objects. In particular, they can be used to represent any of the datatypes permissible in Skeme including numbers, booleans, primitive functions, de Bruijn indices, closures and pairs. However, they can also represent the bytecodes of a compiled Skeme program. We call the set of bytecode actors representing a compiled program, a distributed virtual machine (DVM). Like other objects, a bytecode actor will respond to a get message by returning its value, but unlike actors representing other objects, it can also send and receive encapsulated virtual machine states, or continuations. Upon receipt of a continuation, a bytecode actor transforms it in a manner specific to its type, then passes it on to the next bytecode actor in the program, and so on, until the continuation reaches a halt bytecode. In contrast to a conventional VM, where all control is centralized, control in a DVM is distributed among the bytecodes which comprise it. Furthermore, because Skeme is purely functional, multiple instances of each object and multiple execution threads (continuations) can coexist without inconsistency in the same heap.

Recall that applying a function requires the construction of a stack of evaluated subexpressions. In the simplest case, these subexpressions are constants, and the stack is constructed by executing the *constant* and *argument* bytecodes in alternation. This two bytecode sequence is used to illustrate the operation of a DVM in more detail.

A bytecode actor of type *constant* in the *locked* state loads its accumulator with the address of its operand and enters the

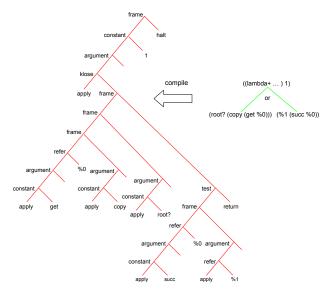


Figure 3: A self-replicating object code program consisting of 37 bytecodes and operands that copies itself by reflection (left) and its Skeme source code (right).

continue state. When an actor in the continue state sees its child in the bytecode tree within its neighborhood, it overwrites the child actor's registers with the contents of its own registers, sets the child actor's state to locked, and returns to the ready state.

The behavior of a bytecode actor of type *argument* in the *locked* state is more complicated. It must push its accumulator onto the argument stack, which is comprised of heap-allocated pairs. Since this requires allocating a new pair, it remains in the *put* state until it sees an adjacent empty site in its neighborhood. After creating the new pair actor on the empty site, it increments the register representing the last allocated heap address (for the execution thread) and enters the *continue* state. See Figure 2.

Self-Replication by Reflection

Since machine language programs are just sequences of values stored in memory, and since instruction sets include instructions which read from and write to memory, it is straightforward to construct a machine language program which copies itself using reflection. Primitive functions providing comparable functionality can be added to Skeme. The most important of these are: copy which makes a copy of a heap-allocated object; root? which returns true if its argument is the last heap-allocated object in an object code program and false otherwise; and get, which given an integer address, returns the heap-allocated object with that address. For convenience, we also add a primitive function succ which returns the successor of its integer argument. Using these functions, it is possible to define a Skeme expression which will copy a set of heap-allocated objects representing the bytecodes and operands of an object code pro-

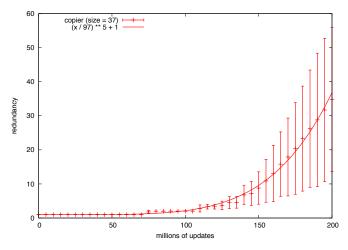


Figure 4: Redundancy versus time for reflection-based SRDVM using diffusion for message passing and best fit function of the form $f(x) = (x/a)^b + 1$. The average is over ten runs. Grid size was 256×256 . Error bars show $\pm \sigma$. The computation becomes more efficient and robust over time.

gram stored in a contiguous range of addresses. This expression compiles into an object code program containing 37 bytecodes and operands (Figure 3). When reified as actors undergoing diffusion, the bytecodes and operands of the compiled program form a *self-replicating DVM*.

Execution begins when the root actor (the actor with largest address among the bytecodes which comprise the program) is sent an initial continuation. When execution ends, two new continuations are launched. To accomplish this, a behavior was added to the actor representing the copy primitive function. When the root actor is copied (the penultimate step in execution), both the root and the copy are sent initial continuations. Because the parent continuation dies a short time later (when it reaches the halt bytecode), the net increase in continuations is one. This ensures that the number of execution threads equals the redundancy increase due to self-replication. However, self-replication alone cannot entirely account for the rate of increase in redundancy observed in Figure 4. Part of the increase is due to the fact that message passing latency in a DVM decreases as redundancy increases (Williams, 2012). Since self-replication time decreases with latency, latency decreases as redundancy increases, and redundancy is increased by self-replication, the process is self-reinforcing. In effect, the actors comprising an SRDVM form an autocatalytic set; as the reactant concentration (redundancy) increases, the reaction rate (efficiency) increases correspondingly.

Robust Self-Assembly of Address Sorted Loops

In this section we describe a method for further increasing the efficiency of message passing in DVMs. This is accomplished by representing the heap as an *address sorted loop*.

In order to implement address sorted loops, the actors

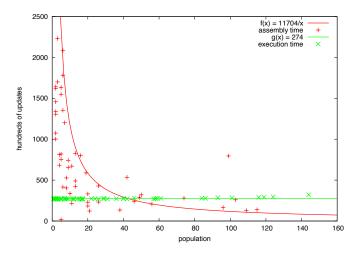


Figure 5: Times required for self-assembly (red) and execution (green) for reflection-based SRDVM as a function of population and best fit functions of the forms f(x) = a/x and g(x) = b. The plotted times are for a single frame bytecode accumulated over ten runs. Grid size was 128×128 . Self-assembly time decreases with increasing population while execution time remains constant.

which comprise DVMs are augmented in several ways. First, they are given a pair of bonds; the *next* bond links an actor to the actor which follows it in the loop, while the *prev* bond links it to the actor which precedes it. Second, every actor is given three additional address registers. The first two (*max* and *min*) hold the maximum and minimum addresses in the loop to which the actor belongs; the third is a *backup* copy of the actor's own address.

The loop is maintained in address sorted order by a low-level behavior implemented by all actors, namely, if an actor's address is greater than the address of the *next* actor (or less than the address of the *prev* actor) then the two actors *swap* positions in the loop. When two actors are swapped, all type and state information is copied from one position to another; bonds are unaffected. The address sorting behavior is modified for the pair of actors with minimum and maximum addresses. These actors do not swap positions but instead serve as anchors for the loop.

A second low-level behavior maintains the maximum and minimum address values. This is accomplished by setting an actor's *max* value equal to the maximum of its neighbors' *max* values (a similar process updates *min*).

Address sorted loops self-assemble by a process which adds one actor at a time in order of increasing address. The self-assembly process is initiated when the actor with address one sees the actor with address two in its neighborhood and forms a length two loop by creating a pair of *prev* and *next* bonds. The *min* and *max* fields of both actors are assigned the values one and two.

When any actor in a loop sees an unbonded actor with address equal to *max* plus one in its neighborhood, it *splices*

that actor into the loop; the new actor is moved to a position midway between the actor doing the splicing and the actor which follows (if there is not enough room the action fails). Appropriate surgeries are then performed on the actor's *next* bond (and the *prev* bond of the actor which follows). In this way, the new actor is incorporated into the loop. The low-level sorting behavior possessed by all actors then rapidly moves it to its correct position.

The current *max* address value is also rapidly updated. Due to the non-zero latency required to update the *max* address value, it occasionally happens that two actors with the same address are added to a loop. To deal with this contingency, yet another low-level behavior is used to *eject* one of any two adjacent actors with duplicate *backup* addresses. This can only happen if the gap which would result from the ejection is short enough to be spanned by a bond; if not, the action fails.

Although we experimented with a more complex method of loop self-assembly involving merging of loops with overlapping address ranges followed by ejection of duplicates, the simple self-assembly method of adding one actor at a time turned out to be the most efficient. This is due to the fact that the net diffusion constant of a set of actors linked by bonds rapidly decreases with the size of the set. By comparison, the diffusion constant of an unbonded actor is enormous. In the simple self-assembly method, the length of the loop is analogous to the surface area of a growing sponge. Like a sponge, it is immobile. However, as the loop grows, it becomes more efficient at collecting an actor with the address it needs to extend itself; the self-assembly process actually speeds up. The time (measured in average number of updates per site) required for loop self-assembly decreases as redundancy increases (Figure 5).

The DVM begins execution when the root actor is spliced into the loop. Message passing is accomplished by a simple process in which the sender temporarily changes its address to the address of the recipient. The low-level sorting behavior then moves the sender to a position in the loop adjacent to the recipient, at which point, the message is delivered. Afterwards, the sender restores its own address (using its *backup* copy) and the low-level sorting behavior rapidly returns it to its original position.

Since swapping two actors joined by a bond does not depend on the relative positions of the actors in the grid (unlike splicing and ejection which fail if the bonds which would result are too short or too long), the message passing process is relatively efficient, taking time proportional to the length of the loop. Experiments will show that the speedup relative to diffusion-based message passing is significant.

Recall that evaluation of a Skeme expression requires three different stacks comprised of pairs that are stored in the heap. In fact, these pairs, which are created by bytecode actors (not by the *cons* primitive function) form the bulk of the heap at any given time. Other *transient* objects, *e.g.*,

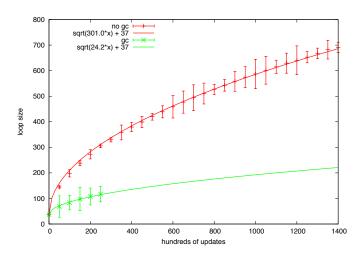


Figure 6: Average loop size versus time for reflection-based SRDVM with (green) and without (red) garbage collection and best fit functions of the form $f(x) = \sqrt{ax} + 37$. The average is over ten runs. Grid size was 128×128 . Error bars show $\pm 10 \sigma$ (for clarity). The SRDVM with garbage collection replicates five times faster.

closures, are also created during evaluation. The fact that message passing latency is proportional to heap size suggests that an aggressive incremental garbage collection process that ejects transient objects from loops when they are no longer needed would yield significant increases in evaluation efficiency.

As part of the implementation of the garbage collection process, the heap's address range is divided into two parts. The first part, consisting of positive addresses, contains permanent objects that form the bytecodes and operands of the mother and daughter object code programs. The second part, consisting of negative addresses, contains transient objects allocated by the DVM during execution. Unbonded actors with negative addresses immediately delete themselves.

When an actor representing a heap-allocated object is created, it is spliced into the loop by bisecting the *next* bond of its creator. Actors created by most primitive functions have positive addresses; actors created by bytecodes (pairs forming the VM's three stacks and closures) have negative addresses. Upon creation, actors with positive addresses are ejected since they duplicate actors already in the DVM; actors with negative addresses are retained. Consequently, during execution, the size of the positive part of the heap stays constant, while the negative part steadily grows. After ejection, actors with positive addresses self-assemble into new address sorted daughter loops.

Recall that the VM's *frame* and *argument* stacks are popped after the application of a primitive function or closure. Whenever a stack is popped, the pairs that comprised the popped record are marked for ejection. Experiments show that this process yields significant increases in eval-

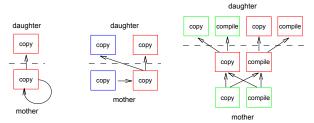


Figure 7: Reflection-based self-replicating program (left) copies itself by exploiting program-data equivalence. Quine self-replicating program (center) with object code genome (blue) and phenome (red). Quine self-replicating program (right) with source code genome (green) and object code phenome (red).

uation efficiency. Indeed, in the case of the reflection-based SRDVM, both the maximum loop size and execution time (measured in average number of updates per site) are five times larger without garbage collection. See Figure 6.

A final behavior only plays a role at the end of execution: when an actor is missing its *next* bond, it deletes its *prev* bond (and vice versa). Since the *halt* bytecode deletes its *next* bond when it receives a continuation, loops rapidly disassemble when the DVMs they host finish executing. Unbonded actors with positive addresses (the bytecodes and operands formerly comprising the mother program) are free to join self-assembling daughter programs; unbounded actors with negative addresses immediately delete themselves.

The life-cycle can be rendered graphically (Williams, 2014). To minimize clutter, only bonds are displayed. The loops' address ranges are mapped to hue so that the effect of the sorting behavior can be verified. The thing that is most striking is the dynamism: everything is moving; nothing is static. SRDVMs look like crumpled rainbows, rapidly assuming different conformations as the actors comprising them undergo diffusion. Continuations move clockwise and counterclockwise inside address-sorted loops that steadily grow, then quickly dissassemble when execution ends.

Self-Replication by Quines

We previously extended Skeme by adding functions that enabled a program to copy heap-allocated objects. In order to define an expression that can recursively copy a binary tree representation of object code, we now extend Skeme with a set of functions that make instances of bytecode types, e.g., the primitive function make-frame to make bytecodes of type frame. For convenience, we also introduce a primitive function, make, which when applied to a bytecode, returns the primitive function that makes instances of that type. Together, these functions make it possible to define an expression that recursively copies object code. One might wonder whether this expression could be compiled and applied to itself, yielding a self-replicating object code program. Alas, this is not possible, since it would require that the program

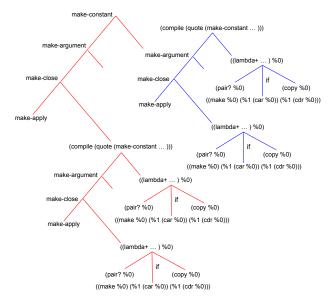


Figure 8: Skeme source code that constructs a self-replicating object code program consisting of 327 bytecodes and operands. Both phenome (red) and genome (blue) are object code programs. Translation and transcription are implemented by identical expressions that copy object code.

contain a *cycle* and cycles cannot be created in pure functional code; Moreover, even if a cycle could be created, there would be no way for the program to know when to stop copying; a more subtle approach is required (Figure 7).

A *quine* is a program that prints itself. All quines consist of two parts. Conventionally called *program* and *data*, they may be thought of as *phenome* and *genome*. All quines work the same way. Active program transforms passive data in two ways producing new instances of both program and data. Equivalently, the *mother* quine's genome is *transcribed* and *translated* yielding the *daughter* quine's genome and phenome. The forms of the genome and phenome, and the nature of the translation and transcription processes, differ from quine to quine. [Hasegawa and McMullin (2013) recently defined a quine inside Avida. To our knowledge, this is the first time this has been done.]

Quines can be written in any programming language but Skeme's list-based syntax, together with quotation, make it easy to write an especially short and simple one. In the following Skeme quine, phenome is an expression that evaluates to a closure that appends a value to the same value quoted; genome is just phenome quoted. Finally, phenome is applied to genome:

```
((lambda (list %0 (list quote %0)))
(quote (lambda (list %0 (list quote %0)))))
```

It is possible to write an object code quine that works much the same way. In place of the special-forms *lambda* and *quote*, the object code quine uses the bytecodes *close* and *constant*. Instead of building a function application

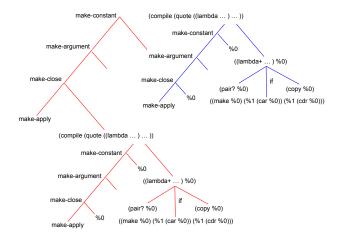


Figure 9: The fact that the translation and transcription processes both return the copied genome can be exploited by introducing a name for this value with a *lambda* expression. The result is a more efficient self-replicating program.

in source code with *list*, the equivalent is built in object code using *make-argument* and *make-apply*. Finally, while the source code quine doesn't actually copy its components (since references suffice for printing) the object code quine must (since the goal is self-replication). The self-similarity of the object code quine can be appreciated by inspecting the Skeme source code that is used to build it (Figure 8). The four bytecodes that comprise the quine's backbone create a closure and apply it to a quoted copy of its own body. These actions build object code that will itself (when executed) create and apply a closure to a quoted copy of its own body; that is, will construct another copy of the quine.

The quine thus constructed contains 327 bytecodes and operands. However, it is needlessly inefficient since it copies its object code genome twice using identical translation and transcription processes. It can be made significantly smaller and more efficient by introducing a name for the value of the copied genome with a *lambda* expression (Figure 9). This quine copies its genome once but uses the copy twice (once as genome and once as phenome) and contains only 107 bytecodes and operands. Its replication rate is contrasted with that of the reflection-based SRDVM in Figure 10.

A self-hosting compiler compiles the same language it is written in. Consequently, it can compile *itself*. It is possible to define a very short self-hosting compiler φ for Skeme (Figure 11). Inserting a copy of φ into the unquoted half of the Skeme quine (phenome) so that it compiles its result and mirroring this change in the quoted half (genome) yields

```
((lambda (\phi (list %0 (list quote %0)))) (quote (lambda (\phi (list %0 (list quote %0))))))
```

which, although not a quine itself, returns a quine when evaluated; this quine is not a source code fixed-point of the Skeme interpreter but an object code fixed-point of the Skeme VM. In effect, it is a quine in a low-level language

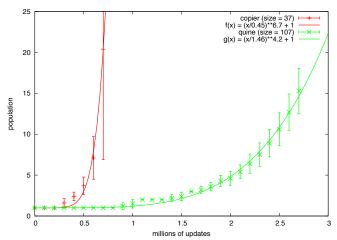


Figure 10: Average population versus time for 37 bytecode reflection-based SRDVM (red) and 107 bytecode quine-based SRDVM (green) and best fit functions of the form $(x/a)^b + 1$. The averages are over ten runs. Grid size was 256×256 . Error bars show $\pm \sigma$.

(phenome) that reproduces by compiling (translation) and copying (transcription) a compressed self-description written in a high-level language (genome). When reified, the SRDVM consists of 990 actors representing a mixture of source and object code. It has been verified that it replicates perfectly across generations and simple statistics including maximum heap and loop sizes, and execution time (measured in average number of updates per site) have been determined for it and for the other SRDVMs. See Table 2.

Table 2: A comparison of four SRDVMs.

code size	max heap size	max loop size	execution time
37	7.2×10^{2}	1.3×10^{2}	2.7×10^{4}
107	1.7×10^{3}	3.7×10^{2}	2.7×10^{5}
327	5.5×10^{3}	1.1×10^{3}	2.2×10^{6}
990	1.8×10^{4}	5.0×10^{3}	2.5×10^{7}

Future Work

Before SRDVMs can be used in the artificial life approach to evolutionary computation several unsolved problems must be addressed. These include the isolation of genomes, reproduction with variation, and self-replication efficiency.

Membranes were essential to the evolution of complex life. Without membranes to concentrate reactants and isolate genomes, neither metabolism nor evolution would be possible. The address sorted loops described here concentrate reactants quite effectively, but do not (currently) provide a mechanism for the isolation of genomes.

If SRDVMs are to evolve, then they must not only isolate their genomes, they must reproduce with variation. Artificial organisms in Tierra and Avida do this because the host sys-

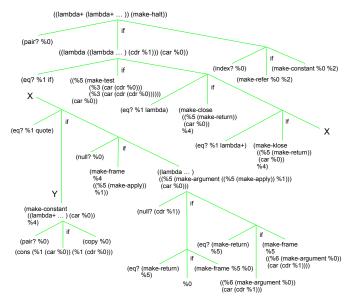


Figure 11: Self-hosting compiler for Skeme can be used to construct an SRDVM comprised of 990 actors of mixed source and object code. The compiler defines the *genotype-phenotype mapping* of a new kind of artificial organism with source code genome and object code phenome.

tems introduce random changes in the instructions of hosted organisms. Spector and Robinson (2002) describe a compelling alternative approach where artificial organisms not only manage their own replication, but also the mutation of their genomes. For SRDVMs to reproduce with variation in this way, a set of mutation operators needs to be defined in Skeme and these operators need to be employed in the subtree of the compiler that copies the genome.

The last problem that must be solved is efficiency. There is nearly a thousandfold difference between the shortest and longest execution times in Table 2. Given that a DVM requires $O(M^2)$ time to build a heap of size M, this difference is understandable. Nevertheless, before they can be useful, SRDVMs must be efficient, and although O(1) message passing latency is not possible in a machine that isn't random access, we believe that $O(\sqrt{M})$ is, and this would be a significant improvement.

Conclusion

The breadth of research in artificial life ranges from genetic programming in Lisp to the engineering of biochemical protocells. It might seem that the gulf between these topics is so large that it will never be spanned. Yet many stepping stones already exist. The concept of *fixed-points in a chemical lambda calculus* described by Fontana and Buss (1994) bridged part of the gulf in the author's own mind several years ago and inspired this paper. By introducing DVMs to the artificial life community and demonstrating an SRDVM that replicates by compiling its own source code, this paper attempts to place another stepping stone in the gulf. It

does so by suggesting that the artificial life approach to evolutionary computation exemplified by systems like Tierra and Avida might be pursued using self-replicating programs written in high-level languages hosted on a computational substrate that has the dual virtues of making competition between programs for all uses of memory zero sum and being indefinitely scalable.

Acknowledgements

The author wishes to thank Dave Ackley, Tom Hayes, and Barry McMullin for helpful conversations.

References

- Ackley, D. (2013). Bespoke physics for living technology. *Artificial Life*, 19(3-4):347–364.
- Adami, C., Brown, C. T., and Kellogg, W. (1994). Evolutionary learning in the 2D artificial life system "Avida". In *Artificial Life IV*, pages 377–381. MIT Press.
- Berman, P. and Simon, J. (1988). Investigations of fault-tolerant networks of computers. In *STOC*, pages 66–77.
- De Bruijn, N. G. (1972). Lambda calculus notation with nameless dummies: a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indagationes Mathematicae*, 34:381–392.
- Dybvig, R. K. (1987). *Three implementation models for Scheme*. PhD thesis, University of North Carolina.
- Fontana, W. and Buss, L. W. (1994). What would be conserved if the tape were played twice? *Proc. Natl. Acad. Sci. USA*, 91:757–761.
- Hasegawa, T. and McMullin, B. (2013). Exploring the pointmutation space of a von Neumann self-reproducer within the Avida world. In ECAL.
- Hewitt, C., Bishop, P., and Steiger, R. (1973). A universal modular actor formalism for artificial intelligence. In *IJCAI*.
- Landin, P. J. (1964). The mechanical evaluation of expressions. *The Computer Journal*, 6(4):308–320.
- Nakamura, K. (1974). Asynchronous cellular automata and their computational ability. System Comput. Controls, 15(5):56– 66.
- Nehaniv, C. L. (2004). Asynchronous automata networks can emulate any synchronous automata network. *IJAC*, 14(5-6):719–739
- Ray, T. S. (1994). An evolutionary approach to synthetic biology, Zen and the art of creating life. *Artificial Life*, 1:179–209.
- Spector, L. and Robinson, A. (2002). Genetic programming and auto-constructive evolution with the Push programming language. *Genetic Programming and Evolvable Machines*, 3(1):7–40.
- Williams, L. R. (2012). Robust evaluation of expressions by distributed virtual machines. In *UCNC*.
- Williams, L. R. (2014). Self-replicating distributed virtual machines (video supplement). In http://www.cs.unm.edu/~williams/srdvm.mov.

Testing the Effects of Speciation and Mutation Rates on Distance-Based Phylogenetic Tree Construction Accuracy Using EcoSim

Ryan Scott¹, Robin Gras¹

¹University of Windsor scotto@uwindsor.ca

Abstract

Biologists regularly construct phylogenetic trees during their research in order to understand the evolutionary history of their subjects. Typically the actual phylogenetic tree is not known, and thus the phylogenetic tree produced is only an estimate. There are two main categories of phylogenetic tree construction, distance-based methods and character-based methods. In all cases, the effects of mutation rate and genetic compactness of species clusters on phylogenetic tree construction accuracy are not well understood. In order to test the correctness of a particular method, it is imperative to perform the study in a system for which the actual phylogenetic tree is known. Thus, realistic and complex simulations in which evolution and speciation occur provide a perfect platform for such a study. EcoSim is such a simulation, and is a simulation in which predator and prey agents interact, evolve, and speciate. Agents in EcoSim possess a complex, evolving, heritable behavioral model which provides meaningful evolution. Here, EcoSim was used as a platform on which to test the effects of mutation rate and speciation threshold on the accuracy of several distance-based phylogenetic tree construction methods. EcoSim has the ability to record all speciation events during a run, therefore we were able to construct the actual phylogenetic trees allowing us to properly compare estimation methods. We created four EcoSim run types: mutation rate increased (MRI), mutation rate decreased (MRD), speciation threshold increased (STI), and speciation threshold decreased (STD). We created five runs of each type, as well as five runs using the standard EcoSim configuration that all lasted 10000 time-steps. At various time-steps throughout each run, we performed Neighbor-Joining, UPGMA, and Fitch-Margoliash tree construction (with and without bootstrapping) on random subsets of 10 species that existed during that time-step, and compared the results to the actual phylogenetic tree using the symmetric distance metric. We found that Neighbor-Joining and Fitch-Margoliash performed nearly equally well, whereas UPGMA performed relatively poorly overall. Further, we found that an increase in speciation rate leads to performance losses in phylogenetic tree construction whereas modifying the mutation rate typically leads to performance gains.

Keywords: ecosystem, individual-based model, distance-based, phylogeny, speciation, mutation rate, speciation threshold.

Introduction

Phylogenetic tree construction is an important topic in biology. It provides many insights regarding the evolutionary history of a set of species, allowing researchers to better understand the organisms we observe today. Phylogenetic trees consist of edges, internal nodes, and external nodes (leaves). Leaves symbolize operational taxonomic units (OTUs), which are the species we can actually observe, and extract data from, with which to construct the phylogenetic tree. The internal nodes represent hypothetical taxonomic units (HTUs), which are hypothetical most recent common ancestors to all species descending from them. Edges are often used to represent the relatedness of two nodes. In practice, phylogenetic trees are often only an estimate of the real phylogenetic tree, because the actual phylogenetic tree is typically not known. Depending on the data available, there are many methods that a researcher could use to construct a phylogenetic tree. These methods fall into two main categories: character-based methods and distance-based methods.

Distance-based methods could use many different data types to perform phylogenetic construction, which includes genetic distance from sequences, immunological data, and any other appropriate distance calculation. There are three very common methods of distance-based phylogenetic tree construction: Neighbor-Joining (NJ) (Saitou and Nei, 1987), Unweighted Pair Group Method with Arithmentic Mean (UPGMA) (Sneath and Sokal, 1973), and Fitch-Margoliash (FM) (Fitch and Margoliash, 1967). Each algorithm makes specific assumptions and has known properties. For instance, UPGMA should produce a correct tree if distance data is ultrametic, that is, rates of evolution are constant among all taxa. In practice, this is rarely the case. The FM and NJ methods perform well when distance data is additive. This is also not typically the case. Each method, by default, generates a single phylogenetic tree for a given distance matrix. UPGMA is the most efficient algorithm of the three, with a time complexity of O(n²) (Murtagh, 1984). NJ performs in O(n³) time (Mailund et al, 2006), and FM runs in O(n⁴) (Lespinats et al, 2011). Euclidean distance between ndimensional points could be used for phylogenetic tree construction using a distance-based method since distance matrices could be constructed from this data. In contrast, character-based methods rely on phylogenetic characters such as behavioral, morphological, genetic, and molecular data to construct trees. In fact, any attribute that is heritable and

varies among the given taxa could potentially be a phylogenetic character (Grandcolas et al, 2001). Further, if necessary, attributes may be discretized in order to produce character states (Wiley and Lieberman, 2011). Character-based methods are typically far more computationally demanding than distance-based methods (Felsenstein, 1988). However, character-based methods often perform better and are thus used more often than distance-based methods in natural studies since transforming character data to distances results in information loss (Felsenstein, 1988). Since we are not dealing with data from an actual biological system and are constructing many trees, we exclusively use distance-based methods for this experiment.

Bootstrapping is a common practice in phylogenetic tree construction (Felsenstein, 1985). In bootstrapping, original data is resampled with character replacement (Felsenstein, 1985). Normally, 100-1000 such resamplings occur, and a new tree is created for each resampling. Bootstrapping is often performed in conjunction with a consensus step during which the many trees created during bootstrapping are combined into one, and the proportion of trees in which each partition is present is also given as output. Thus, the researcher performing bootstrapping and consensus obtains an understanding of which regions of the tree are more likely true, and which regions are possibly erroneous. The final tree uses partitions that are the most represented partitions of all given trees during bootstrapping, and is known as a consensus tree (Felsenstein, 1988). There are several methods of performing consensus, one of which is known as "majority rule extended". Majority rule extended consensus creates a tree using all most represented partitions, and thus the consensus tree is fully resolved (Felsenstein, 2004). That is, all species in the original data set exist in the consensus tree. This is not the case for all consensus methods. There are many ways to calculate the similarity of two trees, however many are situational and many have not been verified. "Symmetric distance" is a tree dissimilarity measure that represents the dissimilarity of topology between two trees (Felsenstein, 2004). It is a very fast distance to calculate, and there is a maximum distance any two trees can possibly have, given the same set of taxa. Thus, it is especially useful because it can be performed numerous times rapidly, and the resulting distances can be normalized by dividing by 2n-6, where n is the number of taxa. Therefore, trees with a normalized symmetric distance (NSD) of 1 share no partitions, and trees with a NSD of 0 are

Given the nature of distance-based phylogenetic tree construction methods, it is reasonable to hypothesize that any factor that affects the positioning and tightness of species clusters in the space of all considered attributes should affect the accuracy of the estimated trees. Mutation rates, for instance, have been known to positively correlate with divergence between species for quite some time (Coyne, 1992; Weir and Schluter, 2008). Thus, mutation rates likely affect the ability to construct accurate phylogenetic trees. Further, an increase in speciation rate causes the emergence of more, smaller species, with consequently less separation between them. With less separation between species, it would

likely be more difficult to properly reconstruct the speciation events of the past.

In this study, we tested the effects of speciation thresholds and mutation rates on distance-based phylogenetic tree construction accuracy. In order to properly test the accuracy of phylogenetic tree construction methods, it is necessary to know the actual phylogenetic tree for the given set of taxa. Furthermore, having the necessary control over the testing environment and the ability to generate repeatable results is essential for any scientific study. A biological study of this scale and depth would be too demanding, in terms of time and resources, to perform. Thus, this type of study lends itself to being performed in a simulation environment. For a simulation to be a sufficient platform for a study of this nature, it must meet several requirements. First, since this is a study on phylogenetic tree construction, the simulation must allow evolution and speciation. Secondly, the simulation must be complex and realistic enough for its results to be valid. Thirdly, the simulation must be able to efficiently produce and store the appropriate data to allow phylogenetic tree construction to occur, and it must also store the series of speciation events that have occurred throughout the course of a run to allow proper comparison. Lastly, as Hang et al (Hang et al, 2007) and Hagstrom et al (Hagstrom et al, 2004) have concluded, natural selection must occur in a simulation environment in order to avoid underestimation of accuracy of phylogenetic tree construction methods. EcoSim is a simulation that satisfied all of these criteria, and therefore EcoSim was employed for this study. We have previously performed a similar study (Scott and Gras, 2012) using EcoSim to test phylogenetic tree construction methods. In the previous study, we only compared the effectiveness of various distance-based phylogenetic tree construction algorithms, without considering various factors that may affect their effectiveness. Here, we look deeper into the factors that affect the construction of phylogenetic trees, using EcoSim as a platform.

EcoSim, an Ecosystem Simulation

EcoSim is an agent-based ecosystem simulation in which predator and prey agents can interact, speciate, and evolve (Gras et al, 2009). Agents in the simulation have a complex and heritable behavior model allowing meaningful evolution to adapt the behaviors of predator and prey over time. The built-in speciation mechanism allows patterns to be analyzed at population, species, or individual scales. EcoSim is an especially interesting simulation because the behaviors of individuals affect both speciation and evolution. A fuzzy cognitive map (FCM) (Kosko, 1986) is used to represent each individual's genomic data which codes for its behavioral model. The FCM contains sensory concepts, internal states, and motor concepts. The FCM is a vector in 364-dimensional real space, where each axis represents the influence of one concept (inhibitory or excitatory) on another. For instance, the sensory concept "partnerClose", which is the perception of a possible sexual partner nearby, would likely decrease the internal states "stress" and "fear" and increase the internal

states "satisfaction" and "sedentary", which should, in turn, increase motor concepts such as "wait", "reproduce", and "socialize". Since the relationships between these concepts evolve throughout the course of a run, the meanings of these concepts tend to change over time. This representation of the genome and behavior model provides not only a consequence and meaningfulness to evolution (that is, natural selection), but also a computationally efficient means of representing a complex concept. As the FCM is heritable, it is mainly responsible for the speciation, evolution, and behavioral models which make EcoSim so realistic, complex, and intriguing. EcoSim subscribes to the "genotypic cluster" species definition (Mallet, 1995). Thus, species in EcoSim naturally form well-defined clusters of points in 364dimensional space. In EcoSim, if any two conspecific individuals are more genetically distant than a pre-defined speciation threshold, the species will split and a new species will arise (Aspinall and Gras, 2010). Every species in EcoSim is identified by an integer value beginning with 1. Thus, species 1 is a common ancestor for all other species in a run. EcoSim tracks all speciation events throughout a run, and therefore we are able to compare estimated phylogenetic trees with a factual tree. This is typically not the case in studies using real biological data. Since genomes in EcoSim are represented by 364-dimensional vectors, we can easily calculate the average genome of a species at any time-step. We can then use this data as input to a distance matrix, and thus we can perform and test any distance-based phylogenetic tree construction methods using data from EcoSim. Further, EcoSim has a robust parameter set and exhibits excellent customizability, thus we can easily modify the speciation threshold and mutation rate while leaving other characteristics of the simulation unaffected. A number of studies have been performed using EcoSim which further validate its use. For instance, EcoSim has been shown to exhibit chaotic behavior with multi-fractal properties (Golestani and Gras, 2010) and realistic species abundance patterns (Devaurs et al, 2010). EcoSim has also been employed in studies of spatial and spatiotemporal distribution (Mashayekhi and Gras, 2012), extinction (Mashayekhi et al, 2014), the importance of predators on regulation of prey populations (Khater et al, 2014), and the effects of physical obstacles on gene flow (Golestani et al, 2012). In all studies, the results produced by EcoSim are consistent with findings from biological experiments.

Data Preparation and Phylogenetic Methods

EcoSim was modified from its original configuration (Normal), with mutation rate of 0.005 and speciation threshold of 3.0, to have its mutation rate parameter increased to 0.006 and decreased to 0.004 (MRI and MRD, respectively), and also to have its speciation threshold parameter increased to 3.5 and decreased to 2.5 (STI and STD, respectively), giving a total of five distinct run types. Five runs of each type were executed, to a maximum of 10000 time-steps. Among all runs, many characteristics differ since

EcoSim is nondeterministic and chaotic. For instance, the average distance between species (standard deviation in parentheses) in the sampled time-steps of the Normal, STD, STI, MRD, and MRI runs were 1.15 (0.24), 0.993 (0.27), 1.25 (0.26), 1.14 (0.20), and 1.29 (0.30) respectively. Further, the average number of species in the sampled time-steps of the Normal, STD, STI, MRD, and MRI runs were 8.04 (3.29), 12.27 (5.94), 6.98 (2.05), 6.31 (2.36), and 8.36 (3.24), respectively. Each run was sampled every 500 time-steps starting at time-step 1500, as each run had enough species (more than 4) to warrant phylogenetic tree construction by this time-step.

It is interesting to consider an unnatural construct such as "speciation threshold" in a study of this sort. Though it is unnatural and a particular speciation threshold is arbitrary, we can still consider the relative effects of a particular threshold. We can consider speciation threshold as a maximum of intraspecific variation for any species. In fact, it is well-known that intraspecific genetic variation changes among species, communities, and populations. For instance, it has been shown in a study by Sides et al (Sides et al, 2014) that genetic variation in a community of related species increases with increased morphological variation.

A program was developed to automatically produce phylogenetic trees for each run in NEWICK format (Felsenstein, 2004) by obtaining data pertaining to species splitting events throughout each run of the simulation. Then, for each sampled time-step, a modified actual phylogenetic tree was produced consisting only of a randomly generated subset of species (of maximum size ten) that existed during that particular time-step. We implemented this limit on the number of species sampled per time-step because in a previous study (Scott and Gras, 2012), we determined that the number of species in a particular phylogenetic tree greatly influenced the accuracy of estimation, despite normalization of tree distances. The identities of the randomly selected species were saved for production of distance matrices (and consequently phylogenetic tree estimations).

From the previously obtained samples, a program was developed to construct distance matrices using the original sample and then bootstrap samples as well, with 1000 resamplings. These distance matrices contained only the randomly chosen species that were previously mentioned, and consisted of Euclidean distances between FCMs of each of these species. After distance matrices were obtained, each tree construction method was performed (UPGMA, NJ, and FM) using "Neighbor" and "Fitch" of PHYLIP (the PHYLogeny Inference Package) (Felsenstein, 1989). In the case of bootstrapping, majority extended consensus was performed using "Consense" of PHYLIP to obtain fully resolved consensus trees. After all of the trees were built, "TreeDist" of PHYLIP was employed to calculate the accuracy of the estimated trees using the aforementioned symmetric distance to measure the distance between estimated and actual trees.

In this study, the entire process of sampling and production of results was automated using scripts and programs that we developed. As mentioned in a previous study (Scott and Gras, 2012), construction of each NJ and UPGMA tree takes less than a second, whereas FM trees can sometimes take up to ten seconds. Despite the fact that an EcoSim run may have to manage hundreds of thousands of "intelligent" agents simultaneously, the time complexity of the system is linear with respect to the number of agents. Thus, we can compute many time-steps in a relatively short time, giving us the ability to observe intriguing evolutionary phenomena and the inception and extinction of many species. For instance, an EcoSim run itself can take roughly a month of total computational time to produce 10000 time-steps, but this is dependent on the number of individuals in the simulation, among other factors such as the capabilities of the computers being used.

Results

For each run, an actual phylogenetic tree consisting of every species produced by the run was prepared. Figure 1 provides an example of such a tree, which was constructed for one of the five STI runs. For each of the sampled time-steps, the aforementioned distance matrices were prepared, along with all necessary trees. Using NSD, we obtained the accuracy of all estimated trees for each sample.

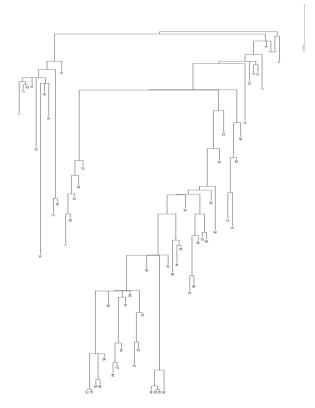


Figure 1: The actual phylogenetic tree for a STI run of EcoSim. The node labels are species IDs that are automatically generated in EcoSim, starting at 1.

Generally, no method performed very well (Figure 2). Here, we did observe many perfect trees (NSD of 0), but we also

observed many trees that are as incorrect as possible (NSD of 1). Though all methods performed poorly, we determined no method performed significantly better (one-way ANOVA, P =0.1101). However, the standard deviation of each set of data was quite large, which will reduce the apparent significance. The standard deviation of these data sets were quite large because each run is unique and can produce many different types of species at any time-step. Overall, we observed that NJ with bootstrapping (NJB) was the best performer, though in this case it performed only marginally better than FM with bootstrapping (FMB). On average, trees produced by NJB exhibited a NSD of 0.511 (0.34), whereas the NSD for trees created by FMB was 0.516 (0.34). UPGMA, on the other hand, performed the worst overall with an average NSD of 0.564 (0.34), which was improved slightly to 0.557 (0.35) when implementing bootstrapping (UPGMAB). In fact, with all methods, bootstrapping slightly improved performance.

Typically, runs in which the mutation rate was modified produced more accurate trees than the original configuration (one-way ANOVA, P=0.0055; Tukey's post hoc test, P<0.05; Figure 2). Normal runs produced trees with an average NSD of 0.518 (0.32). In contrast, MRI runs produced trees with an average NSD of 0.473 (0.33), and the average NSD of trees produced in the MRD runs was 0.477 (.34). In the STI runs, the average NSD was 0.542 (0.32), and in the STD runs we observed an average NSD of 0.645 (0.29), which was the worst overall performance.

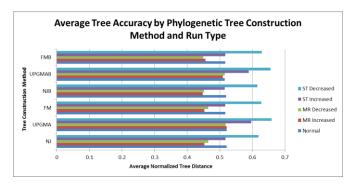
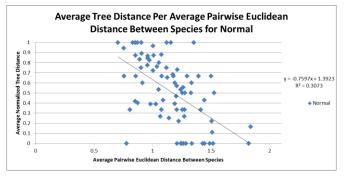


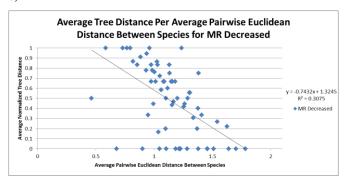
Figure 2: The average normalized tree distance for each run type, grouped by phylogenetic tree construction method. UPGMA generally performed worse than NJ and FM. Modifying the mutation rate leads to better phylogenetic accuracy, whereas decreasing the speciation threshold leads to much worse phylogenetic accuracy.

Since separation between species was a factor we thought may be influential on phylogenetic tree construction accuracy, for every sampled time-step we correlated the average NSD of every tree against the average pairwise Euclidean distance between examined species (Figure 3). Generally speaking, we found a weak negative correlation between average pairwise Euclidean distance between species and average NSD (0.23 < $\rm R^2 < 0.58$). It is observable that increasing the speciation threshold causes increased separation between species, and decreasing the speciation threshold has the opposite effect.

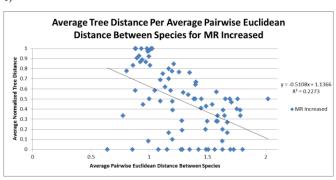
a)



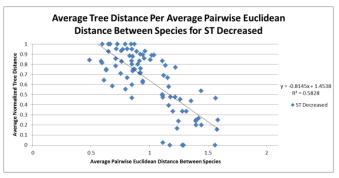
b)



c)



d)



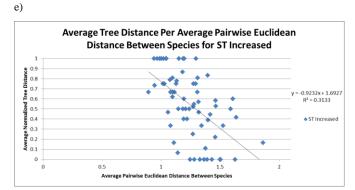


Figure 3: The average normalized symmetric distance per average pairwise genetic distance between species, by run type. In all run types, average normalized symmetric distance seems loosely correlated to average distance between species (0.23 $< R^2 < 0.58$), but there are many other factors that can affect phylogenetic accuracy.

Conclusions

In our experiment, none of the distance-based phylogenetic tree construction methods performed well given the data from EcoSim. On average, all methods were able to correctly reconstruct slightly under half (47.1%) of the partitions in each tree, though trees did range from perfect to completely incorrect. These results are similar from our previous work (Scott and Gras, 2012), in which we found the average tree to contain only 43.5% correct partitions. This is especially interesting because in our previous study we found that an important factor in determining tree accuracy was the number of species, even after normalization. So, although we limit the number of species sampled for each time-step in this study, we still see relatively poor performance of phylogenetic tree construction methods overall. In this study, we found FM, NJ, FMB, and NJB methods to perform very similarly well. This is mostly similar to our previous results, except for the case of NJ, which was the absolute worst performer overall in our previous experiment.

Interestingly, all methods performed best when the mutation rates were modified. Although we observe a correlation between separation between species and tree accuracy for all runs, it is difficult to understand why both decreasing and increasing mutation rates could lead to more accurate phylogenetic trees. The case of increasing mutation rate seems more intuitive, as an increased mutation rate could allow young species to diverge more rapidly from their sister species and thus improve the resolution of the tree. In the case of decreasing mutation rate, however, it may be the case that once a new species is created the species remains in a tighter cluster for a longer time, and because of this accuracy could be increased. The results reflect these concepts. When comparing the results of regression on the MRI and MRD graphs, it is interesting to note the difference in slope and the difference in correlation strength. With the mutation rate decreased, a stronger correlation is observable and NSD decreases more rapidly with increasing average pairwise species distance. In contrast, NSD decreases less rapidly with increasing pairwise distance between species with a weaker correlation when the mutation rate is increased. This indicates that when mutation rate is decreased, more distant species clusters yield more accurate phylogenies. When mutation rate is increased, this is less true, but better phylogenetic accuracy can be achieved with more similar species. With respect to speciation threshold, it is clear that decreasing the speciation threshold reduces phylogenetic accuracy. That is to say that species clusters occurring more tightly in the attribute space yield more inaccurate phylogenies.

We observe a much stronger correlation $(R^2 = 0.58)$ between average pairwise distance between species and phylogenetic accuracy in the STD runs. This is likely because species sets containing very similar species (pairwise distance <= 1) often yield completely incorrect phylogenies. In contrast, for the STI runs, the correlation is weaker $(R^2 =$ 0.31) but performance increases most rapidly with increasing distance (slope = -0.92). This may be because new species are already very distant from their sister species, which can decrease tree resolution. Interestingly, all phylogenetic tree construction methods performed nearly identically with respect to the normal and STI runs, except for UPGMA. In fact, UPGMA performed just as well as the other methods with respect to the normal runs, but exhibited severe performance losses when mutation rates and speciation thresholds were modified. However, the results of this experiment are more congruent with those of a study by Leitner et al (Leitner et al, 1996), which concluded that FM was the most accurate, NJ was of intermediate accuracy, and UPGMA was the least accurate. Further, in both studies, perfect phylogenies were often produced. This study may be more congruent with theirs than our previous study because of the fact that the number of species analyzed in each study is more comparable (they used 9 species, and on average in our previous study we had 29.52).

In the future, we would like to study whether selection of a certain type of attribute (for instance, highly variable or invariable) for use in distance-based phylogenetic tree construction could yield better performance. In this study, we used the entire FCM which may be a reason we observe decreased performance compared to some other studies. It may be the case that overall there are many noisy attributes in the FCM, and some attributes may provide a better phylogenetic signal than others. However, work in phylogenomics (Phylippe et al, 2005; Snel et al, 2005) suggests that using entire genomes increases phylogenetic signal. Further, since some aspects of the effect of mutation rates and speciation thresholds on phylogenetic tree accuracy are still unclear, a logical extension of this study would be to create more run types to see if we can more fully understand the relationships. For instance, instead of having only one bracket of speciation thresholds around the original configuration, it may be better to have three or four brackets (as in a STI1, STI2, STI3, STD1, STD2, STD3) centered about the speciation threshold of the original configuration (and then do the same for mutation rate). Then, we may find that there is some optimal speciation threshold or mutation

rate that yields better phylogenies, as it may be the case that the relationships between phylogenetic accuracy and speciation threshold or mutation rates are parabolic, not linear.

Acknowledgements

This work is supported by the NSERC grant ORGPIN 341854 and the CRC grant 950-2-3617, and is made possible by the facilities of the Shared Hierarchical Academic Research Computing Network (SHARCNET, www.sharcnet.ca).

References

- Aspinall, A., and Gras, R. (2010). K-means clustering as a speciation mechanism within an individual-based evolving predator-prey ecosystem simulation. *Active Media Technology*, pages 318–329, Toronto, Canada.
- Coyne, J. A. (1992). Genetics and Speciation. *Nature*, 355: 511-515. Devaurs, D., and Gras, R. (2010). Species abundance patterns in an ecosystem simulation studied through Fisher's logseries. *Simulation Modelling Practice and Theory*, 18(1), 100-123.
- Felsenstein, J. (1985). Confidence limits on phylogenies: An approach using the bootstrap. *Evolution*, 39:783–791.
- Felsenstein, J. (1988). Phylogenies From Molecular Sequences: Inference and Reliability. *Annual Review of Genetics*, 22:521-565.
- Felsenstein, J. (1989). PHYLIP Phylogeny Inference Package (Version 3.2). Cladistics, 5: 164-166.
- Felsenstein, J. (2004). Inferring Phylogenies. Sinauer Associates, Inc., Sunderland. MA.
- Fitch, W. M. and Margoliash, E. (1967). Construction of phylogenetic trees. *Science*, 155(3760):279-284.
- Golestani, A., and Gras, R. (2010). Regularity analysis of an individual-based ecosystem simulation. *Chaos*, 20(4), 043120.
- Golestani A., Gras R., Cristescu M. (2012). Speciation with gene flow in a heterogeneous virtual world: can physical obstacles accelerate speciation? *Proceedings of the Royal Society B: Biological Sciences*, 279(1740): 3055-3064.
- Grandcolas, P., Deleporte, P., Desutter-Grandcolas, L., and Daugeron, C. (2001). Phylogenetics and Ecology: As Many Characters as Possible Should Be Included in the Cladistic Analysis. *Cladistics*, 17:104-110.
- Gras, R., Devaurs, D., Wozniak, A., and Aspinall, A. (2009). An individual-based evolving predator-prey ecosystem simulation using a fuzzy cognitive map as the behavior model. *Artificial Life*, 15(4), 423-63.
- Hagstrom, G. I., Hang, D. H., Ofria, C., and Torng, E. (2004). Using Avida to Test the Effects of Natural Selection on Phylogenetic Reconstruction Methods. *Artificial Life 10*, pages 157-166. MIT Press, Cambridge, MA.
- Hang, D., Torng, E., Ofria, C., and Schmidt, T. M. (2007). The effect of natural selection on the performance of maximum parsimony. BMC Evolutionary Biology, 7:94.
- Khater, M., Murariu, D., Gras, R. (2014). Contemporary Evolution and Genetic Change of Prey as a Response to Predator Removal. *Ecological Informatics*, 22: 13-22.
- Kosko, B. (1986). Fuzzy cognitive maps. *International journal of man-machine studies*, 24(1), 65-75.
- Leitner, T., Escanilla, D., Franzen, C., Uhlen, M., and Albert, J. (1996). Proceedings of the National Academy of Sciences of the United States of America, 93:10864-10869.
- Lespinats, S., Grando, D., Marechal, E., Hakimi, M., Tenaillon, O., and Bastien, O. (2011). How Fitch-Margoliash Algorithm can Benefit from Multi Dimensional Scaling. *Evolutionary Bioinformatics*, 7:61-85.

- Mailund, T., Brodal, G. S., Fagerberg, R., Pedersen, C. N. S, and Phillips, D. (2006). Recrafting the neighbor-joining method. BMC Bioinformatics, 7:29.
- Mallet, J. (1995). A species definition for the modern synthesis. *Trends in Ecology & Evolution*, 10:294–299.
- Mashayekhi M., Gras R. (2012). Investigating the Effect of Spatial Distribution and Spatiotemporal Information on Speciation using Individual-Based Ecosystem Simulation. *Journal of Computing*, 2(1): 98–103.
- Mashayekhi M., MacPherson B., Gras R. (2014). A machine learning approach to investigate the reasons behind species extinction. *Ecological Informatics*, 20: 58-66.
- Murtagh, F. (1984). Complexities of hierarchic clustering algorithms: state of the art. *Computational Statistic Quarterly*, 1(2):101-113.
- Phylippe, H., Delsuc, F., Brinkmann, H., and Lartillot, N. (2005). Phylogenomics. *Annual Review of Ecology, Evolution, and Systematics*, 36:541-562
- Saitou, N. and Nei, M. (1987). The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology* and Evolution, 4:406-425.
- Scott R., Gras R. (2012). Comparing Distance-Based Phylogenetic Tree Construction Methods Using An Individual-Based Ecosystem Simulation, EcoSim. *Artificial Life 13*, pages 105-110.
- Sides, C. B., Enquist, B. J., Ebersole, J. J., Smith, M. N., Henderson, A. N., Sloat, L. L. (2014). Revisiting Darwin's hypothesis: Does greater intraspecific variability increases species' ecological breadth? *American Journal of Botany*, 101: 56:62.
- Sneath, P. H. A. and Sokal, R. R. (1973). *Numerical Taxonomy*. Freeman, San Fransisco, CA.
- Snel, B., Huynen, M., Dutilh, B. E. (2005). Genome Trees and the Nature of Genome Evolution. Annual Review of Microbiology, 59:191-209.
- Weir, J. T., Schluter, D. (2008). Calibrating the avian molecular clock. Molecular Ecology, 17: 2321–2328.
- Wiley, E. O. and Lieberman, B. S. (2011). *Phylogenetics: Theory and Practice of Phylogenetic Systematics*. John Wiley & Sons, Hoboken, NJ.

Preserving Swarm Identity Over Time

James Stovold¹, Simon O'Keefe¹, and Jon Timmis²

Department of Computer Science Department of Electronics University of York York, UK YO10 5DD jstovold@cs.york.ac.uk

Abstract

Collective identity helps swarms remain coherent in the presence of others. Building identity into artificial systems enables groups of agents to work in the same area as one another, without interference from other agents. By linking the firefly algorithm to the control logic of the agents, we present a method to form and maintain identity in swarms. By measuring swarm polarization, and swarm overlap, we show that the inclusion of an identity allows a swarm to remain coherent for an extended period of time, without interference from other swarms.

Introduction

Collective identity in humans (Melucci, 1995) has been posited as one potential basis for early human survival in the African savannah (Huron, 2001; Jordania, 2011). The ability to focus on a task as a group, rather than acting individually, typically allows a group to complete their task quicker or more effectively. For early humans, this task was typically fending off predators. Introducing collective identity to artificial systems allows groups of agents to remain coherent, while working in the same area as one another. This paper presents a method of forming such an identity, and shows that this identity is able to persist through both time and space.

Melucci (1995) defines identity in a sociological context:

"The term identity...implies the notion of unity, which establishes the limits of a subject and distinguishes it from all others; it implies a relation between two actors that allows their (mutual) recognition." (p. 45)

This definition identifies a number of key characteristics which we can adapt to work with swarms of robots instead of humans. Melucci describes three features that identity always refers to, which can be summarized as:

- Continuity through time
- Separation from others
- Recognition of and by others

Melucci's definition implies that a swarm needs to be able to form and maintain an identity through time (first point), needs to be able to maintain an identity through space (second point), and be able to react to other swarms in the area (third point).

Giving a swarm the means to form an identity allows the swarm to distinguish itself from other swarms, while still being able to recognize and react to other swarms. Following on from the ideas presented in Hofstadter (1979)'s *Ant Fugue*, the identity of the swarm would be at the group or 'signal' level, rather than at the individual or 'ant' level. This means that the identity of a swarm can persist through time and space, resilient to minor changes in the membership of the individuals.

This paper introduces a method of forming emergent identities in a (simulated) swarm of robots, through the use of the firefly algorithm (Tyrrell et al., 2006). We link the synchronization effect of the firefly algorithm to the alignment vector in the boids algorithm (Reynolds, 1987), resulting in small swarms of robots which behave independently of other swarms. We show that these subswarms can remain separated over an extended period of time, even if they encounter other swarms in the area. As two swarms encounter one another, individual robots may switch between swarms, allowing swarms to vary in size while still preserving their identities.

The firefly algorithm offers a method of communicating the swarm membership of an individual, in a way that is visible to any other robots in the area (regardless of their respective memberships). This results in a system where individuals, and in turn swarms, are able to react to each other while maintaining their own identities, and allows swarms to merge and split, forming new identities as required.

The macroscopic behavior of a complex system emerges from the local interactions between comparatively simple, microscopic agents. Because the behavior of a complex system is emergent from these interactions, it is very difficult to engineer systems with specific emergent properties (Stepney et al., 2006). In complex systems, the slightest change in the

underlying algorithm or interactions can give rise to very different macroscopic behavior. By linking the emergent synchronization effects of the firefly algorithm to the control logic of robots, we allow the synchronization of the swarms to influence the behavior of the robots.

Simulated Fireflies The firefly algorithm synchronizes agents as an emergent effect of each agent running a simple procedure. Each agent increments an internal counter up to a predetermined threshold over time. Once it reaches that threshold, the agent flashes a light. If one agent sees another agent flash, it will increment its own counter by a set amount (referred to as *jumping*). As a result, all agents that can see each other for an extended period of time will synchronize to the same frequency (see (Tyrrell et al., 2006) for details).

Simulated Boids In the boids algorithm (Reynolds, 1987), each agent (termed a 'boid') follows three simple rules: alignment, coherence, and separation. At each discrete timestep t, each of the boids calculates its unit velocity vector, \mathbf{v} , for time t+1 by considering those other boids in its immediate vicinity (see fig. 1). If there are any boids in the region of separation, the boid in question will turn away from its closest neighbor. If there are no boids in this region, then the boid in question will align itself to all those boids in the region of alignment by taking the mean of their headings and combining it with an adjustment to turn towards the closest boid in the region of coherence.

We chose to use the boids algorithm as a control algorithm for our agents as it only requires local interactions, in a similar way to the real robotics platform, and offers a cheap method of controlling agents, so that the focus can be on the larger-scale effects of linking a synchronization algorithm with a control algorithm.

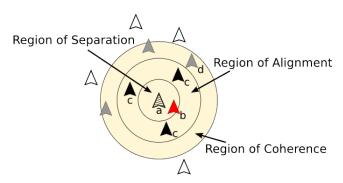


Figure 1: The boid in question (a) turns away from those boids in the region of separation (b), maintaining a set amount of space around them at all times. If boid b is not present, the boid turns towards the average heading of those in the region of alignment (boids c), and turns towards the closest boid in the region of coherence (d).

The Materials and Methods section at the end of the paper

contains the details of its implementation in our simulation, along with details of the measures used to analyze the system.

Results

Our algorithm is able to form multiple swarms from randomly-initialized, simulated agents. These swarms are observed after only 1000 simulated timesteps. The details of the simulation are given in the Materials and Methods section.

Using the simulation, we investigate three specific questions:

RQ1: *Do distinct sub-swarms form and behave independently?*

RQ2: Can sub-swarms preserve their identity for an extended period of time?

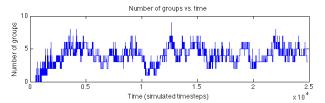
RQ3: Can sub-swarms preserve their identity across space?

Distinct sub-swarms remain highly-polarized through time while global polarization varies

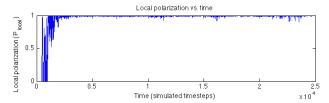
A measure of coherence for sub-swarms is required, which can also be used to measure the global coherence of the swarm. Flock polarization is a measure of the degree of alignment of swarm members (Couzin et al., 2002). We extend the measure to take into account local neighborhoods, to measure the polarization of smaller, distinct swarms.

The polarization of the entire population, P_{global} , is a value between 0 and 1, where 0 implies the vector sum of the headings of all individuals is zero (e.g. two individuals heading east and west), and 1 implies the headings of all individuals are identical (e.g. two individuals both heading west). Our extended metric, P_{local} , measures the polarization within each sub-swarm, such that a value of 0 implies all individuals in each sub-swarm are heading in opposite directions, and a value of 1 implies that all individuals in each sub-swarm are heading the same direction (but not necessarily the same direction as those in another sub-swarm). The full details of these measures are given in the Material and Methods section.

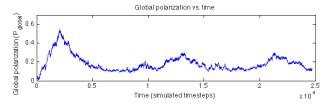
These two measures are compared to determine whether there are large numbers of aligned individuals in few swarms, or fewer aligned individuals in many distinct swarms. Fig. 2 shows the difference between local and global polarization as sub-swarms form, and then repeatedly merge and split. It can be seen that, after the initial swarm formation and settling-down period, the local polarization stays above 0.9 for the remainder of the experiment. This shows that within each swarm the individuals are well-aligned to each other, which is as we expect if the swarms are synchronized.



(a) The number of swarms varies over time as swarms split into subswarms, and merge back into super-swarms.



(b) Local polarization (P_{local}) remains high over the course of the experiment (after the initial settling-down period as swarms initially form). This shows that all swarms in the experiment are coherent.



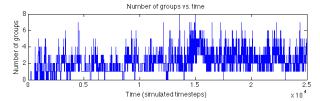
(c) Global polarization (P_{global}) varies over the course of the experiment as the number of swarms (Fig. 2(a)) varies, and remains low throughout. This shows that the swarms that have formed are not all travelling in the same direction.

Figure 2: When the two algorithms are linked, local polarization remains high throughout the experiment and the global polarization varies, but remains low, while swarms split and merge. This shows that sub-swarms can travel in different directions (as shown by P_{global}) while still remaining as coherent sub-swarms (as shown by P_{local}).

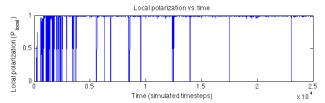
In comparison, the global polarization varies considerably over time but remains predominantly below 0.4 throughout. We can infer from this that there were a number of swarms flocking together, rather than one large swarm, and because those swarms were aligned well to each other (as shown by the local polarization), they were also synchronized with each other.

To confirm that linking the two algorithms is actually having the expected effect, we ran the same polarization experiment with a control case, by removing the link between the two algorithms. The effect of this is shown in Fig. 3, where it is evident that the entire population is consistently heading in a very similar direction throughout, implying zero, or very few, sub-swarms. This confirms that sub-swarms are able to form and, once formed, the sub-swarms are able to head in different directions, while maintaining a high intra-swarm polarization level, implying their behavior

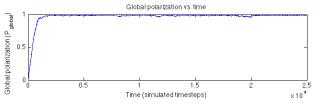
is less dependent on those individuals in other swarms.



(a) The number of swarms varies over time as swarms split into sub-swarms, and merge back into super-swarms.



(b) Local polarization (P_{local}) remains high throughout the course of the experiment. When the number of swarms (Fig. 3(a)) is zero, P_{local} is also set to zero, which can be seen regularly through this experiment.



(c) Global polarization (P_{global}) is consistently high throughout this experiment. This shows that the agents have formed into swarms that are all heading in the same direction, as in the classic boids algorithm (Reynolds, 1987).

Figure 3: When the two algorithms are unlinked, swarms are highly polarized, both globally and locally. This shows that all the swarms in the experiment are heading in approximately the same direction throughout.

Swarms preserve identity over an extended period of time

The swarms formed by the system need to be able to persist through both time and space. To show this, we track specific swarms as they traverse the environment, and observe that they remain intact for an extended period of time, even if they occupy the same area in space as another swarm. The swarms are also resilient to small perturbations in the membership of the swarm (showing that the identity of the swarms are unaffected by individual robots switching swarms). The details of the tracking system are given in the Materials and Methods section.

Fig. 4 shows the length of time each swarm exists for. There are a large number of transient swarms forming (this is a typical effect of the random start position), and fewer, longer-lasting swarms that are able to traverse the environment for an extended period of time (in this case,

up to 6694 simulated timesteps, from a simulation lasting 25000 timesteps).

In the following results, the 'baseline' experiments consist of runs where the two algorithms are linked, and there is no additional noise; the 'noise' experiments consist of runs where there has been additional noise added to the simulation (see Materials and Methods section for details of the noise); and, the 'control' experiments consist of runs where the two algorithms are unlinked.

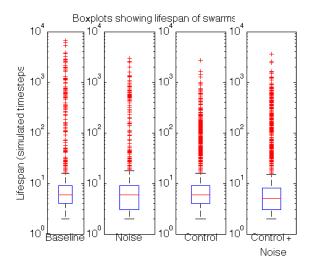


Figure 4: Logarithmic-scale boxplots showing the lifespan of every swarm (≥ 10 agents) for baseline, control, and noisy experimental cases, over runs of 25000 timesteps. Comparing the baseline and control datasets gives p=0.126 and A=0.481; comparing baseline and noise gives p=0.0104 and A=0.533; comparing noise and control + noise gives p=0.0513 and A=0.520; comparing control and control + noise gives p=4.9e-14 and A=0.569. This shows that, statistically, these are from the same distribution. Details of the comparison tests are given in the Materials and Methods section.

The most notable difference between the boxplots in Fig. 4 is that the longest-lasting swarm in the baseline case is 3750 timesteps longer compared with the noise case, and 3991 longer compared with the control case. To confirm that this is typical behavior, we ran the simulation 50 times for each test case (baseline, noise, control, control with noise), and tracked the longest-lasting swarm for each. The results are presented in Fig. 5, where the first two boxplots show the behavior of the system when the two algorithms are linked, and the second two when the algorithms are unlinked (control cases).

Corroborating the previous data, the inclusion of noise in our model reduces the lifespan of the longest-lasting swarm when compared to the baseline case. As anticipated, unlinking the two algorithms—as in the control cases—has a more drastic effect on the lifespan, reducing it much

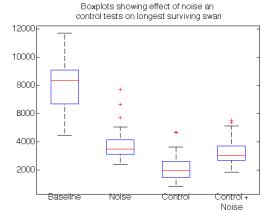


Figure 5: Boxplots showing longest lifespans of swarms over 50 runs. A higher value is indicative of the ability of a swarm to remain cohesive over an extended period of time. The algorithms are linked in the baseline and noise cases, and are unlinked in the control and control + noise cases. Comparing the distributions gives the following results: baseline and noise: p=1.70e-16 and A=0.978; baseline and control: p=8.98e-18 and A=0.998; noise and control + noise: p=0.0019 and A=0.681. This shows that, statistically, these are all from different distributions. Details of the comparison tests are given in the Materials and Methods section.

more significantly compared with the baseline case. When noise is included in the control case, however, we get some unexpected behavior: the inclusion of noise increases the lifespan of swarms when the two algorithms are unlinked.

This increase is due to the increased rate of firefly flashing in noisy, unsynchronized swarms: the tracking algorithm found multiple 'swarms' in each unsynchronized swarm, as when there are sufficient individuals in that swarm which are flashing in synchrony, they will be considered a swarm by the tracking system even though the swarm will also have other individuals which are not flashing in synchrony. This means that when noise is introduced, those individuals who would otherwise change swarms when they are affected by noise, or who would disrupt the synchronization of the swarm (resulting in the swarm breaking up), will now remain in the same swarm, but contribute to a different tracked swarm.

The above results show that the system consistently produces long-lasting swarms, even in the presence of noise. The control case shows that when the two algorithms are unlinked, the identity of a swarm does not persist for any extended period of time (median: 1934.5), compared to the baseline case (median: 8373), and the noisy case (median: 3472), which is as we anticipated.

This shows that the system produces swarms that are able to remain cohesive, and as such preserve their identities, over an extended period of time.

Swarms preserve identity across space

For two swarms to preserve their identities through space, they must be able to occupy the same area in the environment at the same time. To measure this, we calculated the total overlap between swarms at each timestep, to determine the proportion of sub-swarms that were occupying the same area of space at the same time. Overlap calculation details are given in the Materials and Methods section.

Fig. 6(a) shows how the overlap varies as the swarms cross each other over time (for clarity, this is just a smaller section of the 25000 timesteps recorded). The plots in Figs. 6(b) and 6(c) show the overlap from a constructed situation. The situation (Fig. 7) was engineered so that two swarms were already present, and set on paths that coincided with each other. This allowed us to examine the behavior of the swarms without interference from stray individuals or from other groups. The plot in Fig. 6(b) shows the overlap as the two swarms cross each other, with Fig. 6(c) showing the same situation, but with the two algorithms unlinked. Both Figs. 6b and 6c contain three plots pertaining to the median, 5th and 95th percentile of the data over 50 runs.

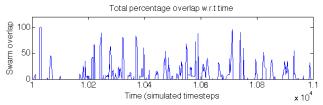
Comparing the two graphs in Figs. 6(b) and 6(c), it is evident that in the control case (6(c)) after the two swarms coincide, the data is more spread out and so the behavior of the model is less well defined in comparison with the baseline case. From this, we can infer that linking the two algorithms together allows the swarms to preserve an identity across space, even if they encounter other swarms.

Discussion

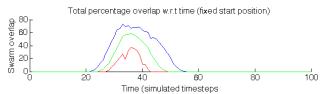
Our results show that, through linking the firefly and boids algorithms, swarms are able to form and maintain identities through both time and space. Giving a swarm an identity gives it the ability to distinguish itself from other swarms in the area. This can help contribute to the ability of swarms to work independently. It also feeds in to the fission-fusion/dynamic swarm problem, in which swarms have to decide when to split, and when to merge, according to the environment they are in. Identity helps swarms to distinguish themselves from others, meaning that we know when one swarm has become two, and vice-versa.

Synchronization methods have been used previously to help guide robots towards a goal. The most notable work in this area (Hauert et al., 2013) considers the problem of guiding flying robots towards a goal using synchronized loitering to prevent them from falling out the sky. This is different from the work presented here, as we are specifically looking at emergent identities, and specifically at how multiple swarms are able to persist in the same environment, even if individuals switch between swarms.

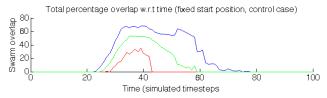
All of the results presented here have been obtained using a specific set of parameters (given in Table 1). The



(a) Plot showing the percentage of total overlap as swarms cross over each other. This is a smaller section of a 25000-timestep run, for clarity.



(b) Plots showing median, 5th, and 95th percentile of overlaps from a fixed start position (over 50 runs). The consistent rise and fall of the overlap over a large number of runs shows that the swarms remain coherent while they pass across each other in space.



(c) Plots showing median, 5th, and 95th percentile of overlaps from a fixed start position (over 50 runs), with the two algorithms unlinked. The consistent rise, but inconsistent fall of the overlap over a large number of runs shows that the swarms are unable to remain coherent when they encounter another swarm in the area.

Figure 6: When the two algorithms are linked, the swarms are able to remain coherent while encountering other swarms in the environment.

sensitivity of the system to these parameters is not known as yet, and is the subject of ongoing work. The most important parameter in the system in terms of scalability is the cycle length of the firefly algorithm (i.e. how high the threshold is for each agent), as this parameter dictates how many different identities can exist simultaneously in the same area, and so imposes a limit on the system. We suspect there is an upper limit, past which the number of swarms that can exist simultaneously in the same area is lower than the cycle length, and so the system is limited by the space taken up by the agents instead of by the algorithm itself.

Further work is needed investigating the effect of the cycle length parameter. The results presented are entirely from computer simulations, the limitations of which are discussed in the Materials and Methods section. Further work would need to include the effect of the presence of physical robots on the algorithm, as physical robots can block the firefly signals.

In summary, we found that the firefly algorithm can be

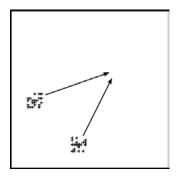


Figure 7: Start position of the fixed-start overlap experiment. The two swarms of 25 agents are randomly distributed within a radius of 4 patches from each start point.

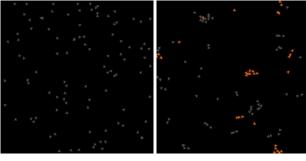
used to influence the control logic of agents, allowing a swarm to form an identity and maintain it through both time and space. The use of swarm identities could allow for the dynamic swarm problem to be approached in novel ways, in particular, disruptions to the identity algorithm would encourage the swarm to re-form identities, giving a window of opportunity for the swarm to split or recombine.

Materials and Methods

Simulation

We simulate the dynamics of the above presented model using NetLogo (Wilensky, 1999). The code used in the simulation was based on two existing simulations of the boids and firefly algorithms (available in the NetLogo models library (Wilensky, 1997, 1998)). We linked the alignment vector in the boids algorithm to the firefly algorithm by defining a new form of neighborhood in each boid: while previously the boids would search for 'flockmates' in their vicinity, and use the headings of those flockmates to calculate a new heading for themselves, the new system introduces 'flashmates' as well. Flashmates are defined as those boids in the immediate vicinity that have flashed at the same time as the boid in question. The proposed algorithm turns each agent towards the mean of its flashmates, rather than the flockmates used in the control system. Fig. 8 shows two screenshots of the system, with Fig. 8(a) showing an initial, random start position, and Fig. 8(b) shows the system after it has been run for 1000 timesteps, and small distinct swarms are clearly visible.

The simulation includes a tracking system that keeps track of which swarm is which between timestep. A swarm is identified in the simulation by taking a boid, then looking at each of its flashmates in turn, and adding them to the group. Then, this process is repeated until all the flashmates of the flashmates have been identified, and the resulting group is considered a single swarm. Once a swarm is identified by the tracking system, it is assigned a numerical group ID, and the number of boids in the swarm is stored



(a) Random start position.

(b) System state after 1000 timesteps.

Figure 8: Screenshots of the simulation, showing the initial, random start position (a), and the state of the system after 1000 timesteps (b), showing small, distinct swarms.

alongside it. In the subsequent timesteps, the number of boids is updated as boids join and leave the swarm. We have included a parameter *group-comparison-threshold* that tells the tracking system how large the percentage change in swarm size can be between timesteps. If the swarm size changes more than this, then the swarm has changed sufficiently to be considered a new swarm, and a new group ID is assigned.

There are a number of assumptions made during the construction of this model. Firstly, the agents in the simulation are modelled as a single point in space; secondly the agents do not have a field of view; thirdly the environment has a periodic boundary.

Of the three assumptions listed, the first will have most effect on the behavior of the system. Having all agents modelled as points means that the firefly algorithm will potentially be able to see more agents in its immediate vicinity than it would be able to otherwise (as they could be blocked by other agents). This concern can, however, be negated by adjusting the firefly algorithm so that it will only jump if it sees another agent, rather than jumping once for each other agent it sees. In addition, there will also be an implication for the boids algorithm, as multiple agents could occupy the same point in space, whereas real robots would block each other if travelling in different directions.

The other two assumptions have a very small impact on the behavior of the system: the 360° visibility is unrealistic in terms of natural systems, but can exist in artificial systems. The periodic boundary on the environment will likely have an impact on how the swarms form in the simulation, but it is unlikely that this will affect the identities of any swarms that do form. This will, however, need confirming in future work.

Another concern when combining these two algorithms is that we need to prevent one of the algorithms taking over and dominating the other. In particular, the firefly algorithm will try to synchronize all the boids in the area, regardless of whether they are in the same swarm. If this happens, the boids algorithm will behave just as it did before. To combat this, we introduce a simple noise term into the firefly algorithm, where one in every 20000 times the algorithm runs, the firefly algorithm will see a robot flashing that isn't there. This type of simple noise is a reasonable form of noise as it helps invert the likelihood of a group of agents to see robots in other swarms (the more robots in the swarm, the fewer robots are left in other swarms). The idea is that if a significantly large number of boids start to form a swarm, they will have a proportionally larger probability of being affected by noise, meaning larger swarms are less likely to form.

A list of relevant parameters, including typical values/ranges is given in table 1.

Parameter	Value
Population	100 (50 in fixed setup)
Max-align-turn	2.50 degrees
Max-cohere-turn	1.75 degrees
Max-separate-turn	1.00 degrees
Cycle-length	4
Flash-length	1
Noise	true / false
Noise-rate	0.00005
Control	true / false
Group-threshold	10
Group-comparison-threshold	90%

Table 1: Parameters used in the simulation, and their typical values.

Measurement

The global polarization measure (Couzin et al., 2002) we use to measure coherence in a swarm was originally defined as:

$$P_{global}(t) = \frac{1}{N} \left| \sum_{i=1}^{N} \mathbf{v}_i(t) \right|$$

where N is the number of agents, and $\mathbf{v}_i(t)$ is the unit velocity vector of agent i at time t.

Because we are interested in smaller, distinct swarms, we extend this measure to work based on local neighborhoods:

$$P_{local}(t) = \frac{1}{K} \sum_{k=1}^{K} \left(\frac{1}{M_k} \left| \Sigma_{j=1}^{M_k} \mathbf{v}_j(t) \right| \right)$$

where K is the number of sub-swarms, M_k is the number of agents in subswarm k, and $\mathbf{v}_j(t)$ is the unit velocity vector of flashmate j at time t. When K is zero (i.e. there are no sub-swarms detected by the tracking system), P_{local} is set to zero (this can be seen in Fig. 3(b)).

To calculate the total overlap at timestep t, O(t), we measure the extent to which the perimeters of each swarm coincides with other perimeters:

$$O(t) = \frac{1}{K} \sum_{i=1}^{K} \left[\sum_{j=i+1}^{K} \left(\frac{o_{ij}}{0.5(P_i + P_j)} * 100 \right) \right]$$

where K is the total number of swarms, P_i is the perimeter of the swarm (calculated as the smallest rectangle that encapsulates the entire swarm), and o_{ij} is the overlap of the perimeters of swarms i and j.

Statistical Comparison

Datasets are compared using two methods, statistical significance and magnitude effect tests. For statistical significance, we use the Mann–Whitney–Wilcoxon U-Test (Mann and Whitney, 1947). For the magnitude effect tests, we use the Vargha–Delaney A-Test (Vargha and Delaney, 2000), which measures the probability that a random sample from one distribution is larger than a random sample from the second distribution. Values for the A test range between 0 and 1, with 0.5 indicating no difference between the two samples and ≥ 0.71 is considered a large difference.

Acknowledgments

James Stovold is funded by an EPSRC Doctoral Training Grant. Jon Timmis is part funded by the Royal Society and the Royal Academy of Engineering. Many thanks to Mark Read for his discussions on the Vargha–Delaney A Test.

References

Couzin, I. D., Krause, J., James, R., Ruxton, G. D., and Franks, N. R. (2002). Collective memory and spatial sorting in animal groups. *Journal of Theoretical Biology*, 218(1):1–11.

Hauert, S., Leven, S., Zufferey, J.-C., and Floreano, D. (2013). Beat-based synchronization and steering for groups of fixed-wing flying robots. In Martinoli, A., Mondada, F., Correll, N., Mermoud, G., Egerstedt, M., Hsieh, M. A., Parker, L. E., and Støy, K., editors, *Distributed Autonomous Robotic Systems*, volume 83 of *Springer Tracts in Advanced Robotics*, pages 281–293. Springer Berlin Heidelberg.

Hofstadter, D. R. (1979). Gödel, Escher, Bach: an Eternal Golden Braid. Random House, New York.

Huron, D. (2001). Is music an evolutionary adaptation? *Annals of the New York Academy of Sciences*, 930(1):43–61.

Jordania, J. (2011). Why Do People Sing?: Music in Human Evolution. Logos.

Mann, H. B. and Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60.

Melucci, A. (1995). *Social Movements and Culture*, chapter 3. University of Minnesota Press.

- Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '87, pages 25–34.
- Stepney, S., Polack, F., and Turner, H. (2006). Engineering emergence. In *ICECCS 2006: 11th IEEE International Conference on Engineering of Complex Computer Systems, Stanford, CA, USA, August 2006*, pages 89–97.
- Tyrrell, A., Auer, G., and Bettstetter, C. (2006). Firefly synchronization in ad hoc networks. In *Proceedings of the MiNEMA Workshop*.
- Vargha, A. and Delaney, H. D. (2000). A critique and improvement of the cl common language effect size statistics of mcgraw and wong. *Journal of Educational and Behavioral Statistics*, 25(2):101–132.
- Wilensky, U. (1997). NetLogo fireflies model. http: //ccl.northwestern.edu/netlogo/models/ Fireflies Center for Connected Learning and Computer-Based Modelling, Northwestern University, Evanston, IL.
- Wilensky, U. (1998). NetLogo flocking model. http://ccl.northwestern.edu/netlogo/models/Flocking Center for Connected Learning and Computer-Based Modelling, Northwestern University, Evanston, IL.
- Wilensky, U. (1999). NetLogo. http://ccl. northwestern.edu/netlogo/ Center for Connected Learning and Computer-Based Modelling, Northwestern University, Evanston, IL.

Optimizing the crossregulation model for scalable abnormality detection

Danesh Tarapore^{1,2,3}, Pedro U. Lima², Jorge Carneiro³ and Anders Lyhne Christensen⁴

¹ ISIR, Université Pierre et Marie Curie 6, CNRS UMR 7222, F-75252, Paris Cedex 05, France. ² Instituto de Sistemas e Robótica, Instituto Superior Técnico, 1049-001 Lisbon, Portugal.

The engineering of fault-detection systems for multirobot systems (MRS) is a well-studied problem (e.g, Christensen et al. (2009)). Most fault-detection models are built on the assumption that normal behavior is known, and can be characterized in advance. The models are trained to recognize predefined normal behaviors, and behaviors not recognized are labeled abnormal. While such an approach does provide some interesting results of robust fault detection and fault tolerance, they may not be applicable when normal behavior can change as a result of unforeseen environmental conditions and online learning for instance. Furthermore, prior information required to characterize normal behaviors, may not always be available.

The adaptive immune system in vertebrates has to allow the body's cells and tissues to function normally, while mounting a response against possible abnormalities (e.g., cancerous cells) (Leon et al., 2003). The characteristics of these abnormalities are in principle open-ended and therefore differ from current approaches to fault detection in robots. The crossregulation model (CRM) (Leon et al., 2003) captures this robust maintenance of immunological tolerance, allowing the system to discriminate between antigens based solely on their density and persistence in the environment. In our previous work (Tarapore et al., 2013), the CRM has been used to develop a decentralized abnormalitydetection system for a MRS, wherein each robot simulates its own private population of virtual cells of the immune system. The system was able to tolerate normal swarm behaviors, characterized as persistent and abundant, while mounting an immune response against abnormal behaving agents. Despite these salient features, the model incurred a severe computational cost, consequent to the high-dimension space of behavioral features in which the the normal and abnormal behaviors were classified. Here, we propose optimizations to the CRM, that allow for an improved scalability in terms of the number of features used for behavior classification.

The CRM describes the population dynamics of cells of the adaptive immune system, consisting of three cell types: (i) antigen presenting cells (APCs) that present the antigen on their surface. Individual APCs have a fixed number of conjugation sites (s) on which T-cells can conjugate; (ii) effector T-cells (T_E) that can potentially mount immune responses which, depending on receptor specificity, may be directed to abnormal foreign pathogens or to normal bodyantigens; and (iii) regulatory T-cells (T_R) that suppress proliferation of T_E cells with similar specificities. A mathematical formulation of T-cell–APC interaction dynamics is detailed in Tarapore et al. (2012). Below, we highlight these interactions, and then describe the optimizations to the model.

The CRM provides a system of differential equations governing the density of each of the clonal types (i) of T_E (E_i) , and T_R (R_i) T-cells. The subpopulations of each of these clonal types is subject to the following: (i) growth by proliferation (division of parent cells into two daughter cells) of their individual activated cells; and (ii) shrinkage consequent to cell death (see parameters in Table 1).

The density of activated $\mathbf{E_i}$ and $\mathbf{R_i}$ cells of each clonal type i, is dependent on their interactions with APCs A_j of each subpopulations j. The resulting T-cell–APC conjugates C_{ij} is described by the following equation:

$$\dot{C}_{ij} = \gamma_c \theta_{ij} (T_i - \sum_{j=1}^M C_{ij}) (A_j s - \sum_{i=1}^N C_{ij}) - \gamma_d C_{ij} \quad (1)$$

where $T_i = E_i + R_i$, and γ_c and γ_d involve the conjugation and deconjugation rates between APCs and T-cells, respectively (parameters in Table 1), and θ_{ij} is the affinity of the interactions between T_i and A_j . We integrate at each time step, the steady state values of the conjugates.

The density of activated T_E and T_R cells is computed from the quasi-steady state densities of the conjugates. The conjugated T_E cells are activated in the absence of T_R cells on the same APC. In contrast, conjugated T_R cells can only be activated if at at least one T_E cell is simultaneously conjugated to the same APC.

The implementation of the CRM is optimised by simplifying the dynamics of the conjugates, assuming that the total T-cell density is in excess of the density of conjugated cells.

$$\dot{C}_j = \gamma_c(\sum_{i=1}^N \theta_{ij} T_i) (A_j s - C_j) - \gamma_d C_j \tag{2}$$

From eq. 2, the quasi-steady state density of the conjugated cells is calculated as the following function, for each

³ Instituto Gulbenkian de Ciência, 2780-156 Oeiras, Portugal.

⁴ Instituto de Telecomunicações, Instituto Universitário de Lisboa (ISCTE-IUL), 1649-026 Lisbon, Portugal. daneshtarapore@gmail.com

Table 1: Parameters of the crossregulation model.

Param.	Description	Value (a.u.)
l	Length of binary feature vector	l
N	Maximum number of T-cell clones	2^l
M	Maximum number of APC subpopulations	2^l
A_j	Density of APCs of population j	_
s	Maximum number of conjugation sites on APC	3
E_i	Density of effector cells of clone i	_
R_i	Density of regulatory cells of clone i	_
T_i	Density of T-cells of clone i	$E_i + R_i$
C_{ij}	Density of conjugates between T_i and A_j	_
γ_c	Conjugation rate of T-cells to APCs	10^{-1}
γ_d	Deconjugation rate of T-cells from APCs	10^{-1}
π_E	Proliferation rate of effector cells	10^{-3}
π_R	Proliferation rate of regulatory cells	0.7×10^{-3}
δ	Death rate of effector and regulatory cells	10^{-6}

existing APC subpopulation j:

$$C_j = (\gamma_c A_j s \sum_{i=1}^N \theta_{ij} T_i) / (\gamma_d + \gamma_c \sum_{i=1}^N \theta_{ij} T_i)$$

The total number of conjugated effector and regulatory cells on the APC subpopulation j is then calculated, proportional to the relative frequency of T_E and T_R cells, and weighted by their affinity to the APC subpopulation j. For the conjugated effector Ee_j and regulatory Re_j cells at APC subpopulation j, we have:

$$Ec_j = C_j \frac{\sum_{i=1}^N \theta_{ij} E_i}{\sum_{i=1}^N \theta_{ij} T_i} \quad \text{and} \quad Rc_j = C_j \frac{\sum_{i=1}^N \theta_{ij} R_i}{\sum_{i=1}^N \theta_{ij} T_i}$$

Finally, the density of activated effector regulatory cells is computed (as in the unoptimized model (Tarapore et al., 2012)), and the population of T_E and T_R cells is updated.

The unoptimized and optimized CRM are implemented on a distributed embodied MRS of 20 e-puck-like robots, so as to detect fault-simulating agents, while maintaining tolerance towards normal swarm behavior. Behaviors that are abundant (performed by most agents) are to be tolerated, and rare behaviors (exhibited by fewer agents) are to be detected as abnormal.

Individual features of an agent's behavior are encoded in Boolean form, and then concatenated to form a binary string, the *feature vector* (FV). A FV comprises 3, 6, 9, 12 and 15 features (Tarapore et al., 2014). At every control cycle, each agent senses the FVs of its ten nearest neighbors, and computes the number of agents assigned to each FV. In the agent's internal CRM instance, APCs are generated corresponding to each of the feature vectors perceived, and the CRM is then executed to determine their status.

Normal behaviors exhibited by the swarm are aggregation, dispersion, flocking and homing towards a moving landmark. The fault-simulating behaviors performed are, (i) move continually in a straightly line; (ii) perform a random walk; (iii) circle around a fixed point; or (iv) stop completely. These behaviors mimic faults such as, software bugs, sensor faults, motor malfunctions and a broken battery. The CRM was able to detect abnormalities in almost all

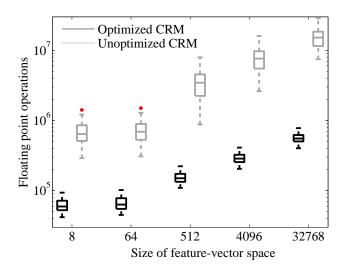


Figure 1: Computational costs of unoptimized (light) and optimized (dark) abnormality detection using the CRM with FV space for 3, 6, 9, 12 and 15-bits FV length, across 20 replicates. Each box corresponds to the average number of FLOs per agent, per control cycle.

normal/fault-simulating behavior combinations (Tarapore et al., 2014). In one such experiment one fault-simulating agent performed random walk, and the other agents of the swarm performed aggregation. We show the number of floating points operations (FLOs) executed by the CRM (Fig. 1). Results indicate that the computational cost increases with an increase in FV length, for both the unoptimized and the optimized models. However, optimizations to the model yield a reduction in execution cost, particularly for large FVs (up to an order of magnitude reduction in FLOs for 15-bit FVs). The results encourage us to use the optimized version of the model in more complex scenarios.

References

Christensen, A. L., O'Grady, R., and Dorigo, M. (2009). From fireflies to fault tolerant swarms of robots. *IEEE Trans. Evol. Comput.*, 13(4):1–12.

Leon, K., Lage, A., and Carneiro, J. (2003). Tolerance and immunity in a mathematical model of T-cell mediated suppression. *J. Theor. Biol.*, 225:107–126.

Tarapore, D., Christensen, A. L., Lima, P. U., and Carneiro, J. (2012). Environment classification in multiagent systems inspired by the adaptive immune system. In *Proc. of ALIFE*, pages 275–282. MIT Press, Cambridge, MA.

Tarapore, D., Christensen, A. L., Lima, P. U., and Carneiro, J. (2013). Abnormality detection in multiagent systems inspired by the adaptive immune system. In *Proc. of AAMAS*, pages 23–30. IFAAMAS, Red Hook, NY.

Tarapore, D., Lima, P. U., Carneiro, J., and Christensen, A. L. (Submitted, 2014). Scalable abnormality detection in multiagent systems.

The Effect of Social Learning on Individual Learning and Evolution

Chris Marriott¹ and Jobran Chebib²

 ¹University of Washington, Tacoma, WA, USA 98402
 ²University of Calgary, Calgary, AB, Canada T2N 1N4 dr.chris.marriott@gmail.com

Abstract

We consider the effects of social learning on the individual learning and genetic evolution of a colony of artificial agents capable of genetic, individual and social modes of adaptation. We confirm that there is strong selection pressure to acquire traits of individual learning and social learning when these are adaptive traits. We show that selection pressure for learning of either kind can supress selection pressure for reproduction or greater fitness. We show that social learning differs from individual learning in that it can support a second evolutionary system that is decoupled from the biological evolutionary system. This decoupling leads to an emergent interaction where immature agents are more likely to engage in learning activities than mature agents.

Introduction

When agents possess both genetic adaptation operating on a generational time scale and learning operating on the agent's time scale there is the potential for interaction between these adaptive mechanisms. Sznajder et al. (2012) have surveyed studies into this interaction and show conditions under which the presence of learning can accelerate or decelerate genetic adaptation.

In particular Paenke et al. (2007) found that if the fitness function in the direction of adaptation is concave and the step size of the learning algorithm is small genetic adaptation can occur at an increased rate. This acceleratory effect was seen in a number of different simulations (for instance Hinton and Nowlan (1987); Fontanari and Meir (1990); Mayley (1997); Lande (2009)).

On the other hand if this condition is not upheld, or indeed the opposite conditions exists, then the learning can slow the natural genetic adaptation (see Papaj (1994); Anderson (1995); Dopazo et al. (2000); Borenstein et al. (2006)). Some simulations have shown both acceleratory and deceleratory effects under different conditions (Ancel (2000); Paenke et al. (2006, 2007)).

Social learning is a form of learning that arises from social situatedness (Lindblom and Ziemke (2003)) and is characterized by agents interacting with one another in order to learn. Social learning can accelerate learning beyond that of

individual learning strategies (see Denaro and Parisi (1997); Acerbi and Parisi (2006); Marriott et al. (2010)) and most notably is its ability to support cumulative cultural evolution (Mesoudi et al. (2006)) as witnessed in human culture. While a wide range of mechanisms of learning have been studied in conjunction with evolution we have found little evidence of the study of social learning mechanisms on evolution. Further, since social learning is a distinct form of learning identified by its social as opposed to individual nature, it may also interact with the individual learning processes. In this study we explore the interactions between social learning, individual learning and genetic adaptation.

We have designed a model that allows for evolution of reproductive, individual learning and social learning abilities in conjunction with traits for fitness. These abilities are not just on/off for our agents but instead can be participated in more or less than others. The abilities also come at a cost to the agent (in time) which limits the maximum fitness achievable by the agent.

The model we have developed allows for the genetic information and the learned information to be expressed in the same format, allowing for direct comparison between genetic adaptation and the individual and social learning processes.

Experimental Setup

Learning Task Our simulation involves agents that gather resources from a number of different resource sites. Each site has five locations where resources might be found. Only one, two, or three of the locations actually have resources while the others are empty. We call this value the reward of the site. Say an agent is at a resource site with three resources hidden in the five locations. The agent must select the order it will check the locations. Once the agent has all three resources it can stop checking, and the cost of this gather attempt is the number of locations checked. This cost is measured in time units.

Agents will have to select which sites to gather from each day and what order to check locations at those sites. The resource sites may not be repeated in a day, but can be repeated again the next day. The next day the resource site will have the same number of resources and they will be stored at the same locations. This allows for the agent to adapt its strategy on future days.

More formally we can think of a resource site as a subset of $\{1,2,3,4,5\}$ with size equal to the reward. For instance with reward equal to 3 the resource site might be the set $\{1,2,4\}$. An attempt at this site by the agent is characterized by a permutation of the set $\{1,2,3,4,5\}$, for instance, [5,4,2,1,3]. The cost to the agent (in time units) at this site is equal to the number of entries in the permutation that must be processed before the subset representing the site has been covered by the elements of the permutation. In our example it requires the first 4 elements of [5,4,2,1,3] to cover the set $\{1,2,4\}$ resulting in a cost of 4 time units spent to gather 3 resources. Notice the maximum cost is always 5 and the minimum cost is equal to the reward.

An agent is allocated 50 time units in a day and may gather from as many sites as it has time units. Thus optimally an agent can gather 50 rewards from resource sites in one day if all time units are dedicated to gathering. An agent may also spend its time units on activities that may increase its long term fitness. Specifically it may spend units attempting to reproduce, learn individually, or learn socially. These tasks each take one time unit and may be taken more than once in a day.

Agent Design The agent consists of a genome and a memome. The genome and the memome are internally identical, but play different roles in the agent. The genome is inert information that remains static for the duration of the agent's life. It is used during asexual reproduction to create a new agent. The memome is dynamic and may change over the lifetime of the agent. It is used to select daily behavior and may be spread socially.

Genomes and memomes are internally identical and in this paragraph we will describe these internals from the perspective of a genome. However, everything is equally true of the memome. A genome consists of 50 geneplexes, each describing a series of actions that could be carried out by an agent in a single day. A geneplex's total cost is the time cost of all actions in the geneplex and is always less than or equal to 50. A geneplex's fitness is equal to the total reward that its actions accrue (ties are broken favoring geneplexes that spend less time attaining the same total reward or spend some time reproducing or learning). When a genome is mutated the five geneplexes with greatest fitness replace the five geneplexes with least fitness and the copies are mutated. A geneplex mutates by changing the strategy at a randomly selected resource site, adding a randomly generated action, or removing a randomly selected action.

At birth an agent will copy its genome to create its memome. When a day's activities are to be selected the agent takes the memeplex with highest fitness and carries out its list of actions. Its daily reward is the total number of resources these actions accrue.

A new agent is created from a parent selected randomly but not uniformly from the population. An agent's chance of being selected is equal to the number of breeding tickets it has. It is rewarded one breeding ticket for every five resources it gathered today and one for each time unit the agent spent on reproducing today. So agents better at gathering and who spend more time on reproduction are more likely to be a parent. The new agent's genome is a mutated clone of its parent's genome.

When an agent is born the agent with least fitness dies to maintain a constant population size. An agent's fitness is equal to the average daily reward during its life minus its age. As a result agents that die are either poorer at gathering, older, or both.

During a day an agent may spend time learning. For each occurrence of an individual learning action in its daily action list the agent will mutate its memome once. This activity will change the memome but leaves the genome unchanged. The new memome is used to select the next day's actions.

An agent is selected for participation in a social gathering with probability proportional to the number of time units spent on socially learning. The selected agents then contribute their best five memeplexes into a pool. The best five in the pool are then redistributed to the agents replacing their five least fit memeplexes. Again this exchange affects only the memome leaving the genome unchanged.

Population Design We considered three isolated populations of 50 agents. Each agent in the initial population has a genome consisting of geneplexes with only a single random gathering action. No genome was seeded with reproducing, individual learning, or social learning actions.

The first population served as a control and agents were not capable of learning in any way. Their only means of behavioral adaptation is genetic evolution. Any actions spent by these agents on learning were wasted actions as they had no result and thus these actions were slightly maladaptive for these agents. The memome and genome were identical in these agents since their memome was incapable of change.

Agents in the second population were capable of learning from the environment but not from each other. Any actions spent on learning from others were wasted actions in this population and thus were slightly maladaptive. Agents had to evolve to learn before they would begin learning.

Agents in the third population were capable of all the learning strategies, and like the agents in the second population, they needed to evolve to learn individually or socially. These learning capabilities were separate and thus evolving one did not imply evolving the other.

We gathered data on the maximum fitness of agents in each population over time. This allowed us to track the optimization occurring in each population. We also tracked the maximum fitness of geneplexes in the genomes of agents in each population. This allowed us to isolate and track genetic optimization. We also gathered data on the frequency of actions for reproducing, individual learning, and social learning among our agents. We selected an epoch length of 10000 days with data samples every 20 days. We have averaged results over 159 runs for presentation. With such a large number of runs the confidence intervals are very small and are omitted for clarity.

Observations and Discussion

Overall Fitness By day 1500 the average maximum memetic fitness of agents in the social learning group increases to a maximum of 48 (see Figure 1). Agents cannot both attain the optimum and maintain their social learning behavior. In most runs agents converged on a shared memeplex with one social learning action and one individual learning action (and occasionally a reproduction action).

This behavior can be considered optimal in that a memeplex of higher fitness would have to replace either the social learning action or the individual learning action with a gathering action, and thus lose the benefit these actions give. Replacing the social learning action might make the agent more fit, but would prevent that action from being shared with others and the more fit memeplex would die with the agent. Replacing the individual learning action would stop individual optimization of the memeplex making it very unlikely to increase fitness. Unlikely as this event is in our model we do observe this occurring infrequently.

By day 1500 the average maximum fitness of agents from the individual learning group has increased to 45. By the end of the 10000 day epoch this has been further optimized to nearly 47.

At the 10000 day epoch individual learners have an average of four actions allocated to reproducing and learning in the memome. Individual learners are responsible for optimizing their own memeplexes whereas social learners can rely on the community to communicate the optimums to them. As a result the individual learners must maintain greater commitment to learning even when near the optimum fitness to be competitive.

The average maximum fitness of agents from the non-learning group increases to only 34 by day 1500. At the end of the 10000 day epoch the average fitness is nearly 44. This is still a few steps from the optimal and this is because we see at the end of the epoch the maximal memeplex had on average four reproduction actions (and one of the null actions).

This behavior should be seen as nearly optimal. Breeding tickets are awarded one for every five rewards achieved and one for every reproduction action. This would suggest that there is an equilibrium between selection pressures rewarding greater fitness and those rewarding time spent on reproduction. Since on average five time units are spent on non-

gathering actions the fitness of these agents can theoretically be increased by five. However this would award the agents only a single additional breeding ticket (in exchange for four lost), greatly decreasing their odds of reproducing. On the other hand agents that spend much more time on breeding actions than four (or five) will reduce their fitness making them more vulnerable to early death due to low fitness. The midpoint between these opposing selective forces appears to be at five time units.

These average maximum fitness results support a common hypothesis regarding the rate of optimization utilizing these different modes of learning (Denaro and Parisi (1997); Acerbi and Parisi (2006); Marriott et al. (2010)). Specifically, genetic optimization is seen as slower than optimization through individual learning or learning from the social group. Furthermore, learning from the social group can maintain learned strategies in the memosphere and thus carry out cummulative learning. This leads to the hypothesis that social learning is the most rapid mode of optimization.

Genetic Fitness Our interest extends further to the effect that these modes of learning have on the genetics of the agents using them. Modern hypotheses (Sznajder et al. (2012)) suggest that the learning mechanisms can both heighten selection, thus accelerating genetic adaptation, and shield it, thus decelerating genetic adaptation.

Both the individual learners and the social learners had less fit genomes compared to the control non-learning group for most of the simulation. At day 1500 both learning groups have a average maximum genome fitness of 30 while the non-learning groups was 34 (see Figure 1). This indicates that the selection pressure operating on the genomes according to fitness levels was shielded by the learning of both kinds in the early days of the simulation. This is predicted in instances like ours where the learning can operate to flatten the fitness landscape.

Optimization is shielded for fitness levels in the range 15 to 30 during an "easy task" of growing the initially short geneplexes to ones that use all 50 time units. After this point the optimization problem increases in difficulty since the geneplexes then can only be optimized by improving the strategies of individual gathering attempts.

There does not appear to be any further shielding effect on the social learners during the "difficult task". That is the slope of improvement for non-learners and social learners is very close after day 1500 (both improve 10 fitness in 8500 days). In contrast the individual learners appear to optimize at a more rapid rate (12 fitness in 8500 days).

In individual learners differences between agents' average daily fitnesses becomes dependent on early performance and speed of convergence while solving the "difficult task". Agents that have less fit genomes perform worse early on and those that take longer to converge spend more time at lower reward levels. Thus, conditions seem met for an ac-

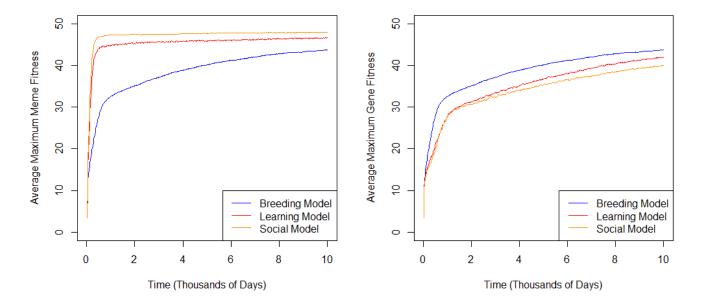


Figure 1: Average maximum fitness of memes (left) and genes (right) in breeders, learners and socializers.

celeration of adaptation of both the genes for learning and the genes for more efficient gathering.

That is, agents that have a more fit genome or occupy positions that allow them to more rapidly overcome genome deficiencies are selected for in the population. On the one hand this pressure rewards agents with more fit genomes because they both perform better early on and can optimize more rapidly from this better position. On the other hand this pressure would favor agents that can learn rapidly indicating selection for individual learning actions (see discussion below). This seems to be a case of Baldwin's initial hypothesis where the selection pressure accelerates adaptation of plasticity simultaneously with greater fitness.

Noting this we might expect also that social learners should benefit from this same acceleration though we did not find this in our data. We did find strong evidence of increased selection for social learning actions (see discussion below) in these agents and we hypothesize the benefits of these actions greatly outweigh benefits to early performance.

Reproductive Actions Non-learning agents converged on an average of four reproduction actions by the end of the epoch. This is due to the selection pressure that is applied related to the increased chance of reproduction for these actions. We expect the same selection pressure to exist for the learning agents as well and we see evidence for that in the genomes of the learners (see Figure 2). The individual learners steadily increase to an average of about 2.75 reproductive actions per agent. Notably this average is lower than

that in the non-learners indicating that this selection pressure is shielded by the learning. We also see evidence of this shielding effect in the social learners who increase to an average of about 1.6 reproductive actions per agent.

A contributing factor to the shielding effect is that the memome is used to determine the number of breeding tickets issued, not the genome. As a result the number of reproductive actions in the genome do not contribute long term to the number of breeding tickets issued to the agent, as the memome is assumed to change from its initial state in learners. As such the reproductive actions in the genome contribute most to the chance of breeding early in life (before learning potentially replaces these actions). This also means that an early genetically determined reproductive behavior can be learned away. This provides fewer opportunities for more fecund genomes to express their greater fecundity resulting in a shielded selection pressure.

We see that in individual learners the number of reproduction actions in the memome is coupled to the number in the genome. While prior to day 3000 there are more learning actions in the memome than the genome, this changes after about day 3000. The reason for this change is as the genetic fitness approaches the optimum the pressure to learn decreases.

In the social learners we see a different dynamic. The average reproductive actions in the memome seem uncoupled from the genome. The reproductive actions in the genome increase due to pressure (if shielded) for increased reproduction. In the memome we have a depreciating curve, indi-

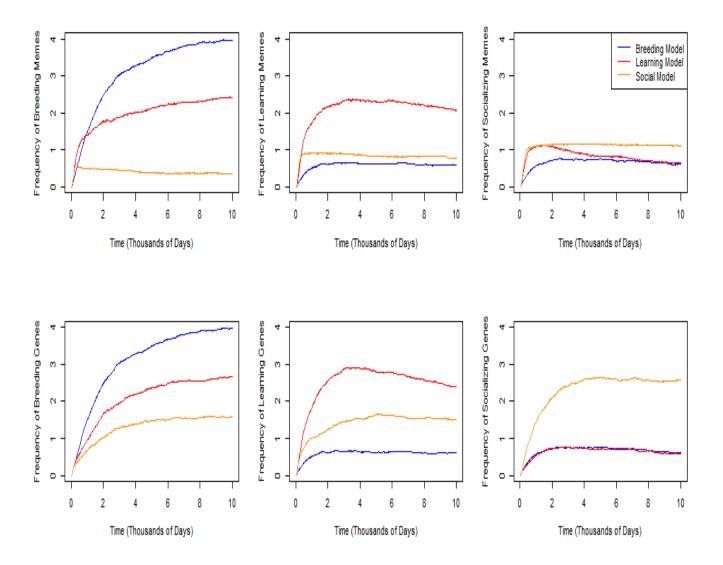


Figure 2: Average number of breeding, learning and socializing actions in the genomes and the memomes of breeders, learners and socializers.

cating selection pressure against breeding in the memome. Breeding actions do not help spread the memeplex, but they do help spread the geneplex, and so we see divergent selection pressures here and a decoupling of the adaptive processes.

This decoupling is due to the independence of the evolution of memeplexes from the evolution of geneplexes and this independence results in cumulative cultural evolution. The individual learning of each individual learner must begin optimizing from the initial condition coded in its genome (hence a coupling of these processes). The social learner in contrast can begin from whatever state is provided by its community. Thus it need not "reinvent the wheel" and re-

peat learning steps that others have already carried out. This leads to a memosphere whose evolution is independent of (decoupled from) the co-evolving genosphere. As a result the selection pressure that occurs in the genosphere for reproductive actions does not exist in the memosphere. This may also be the case for individual learners, but only the social learners have decoupled the memosphere's evolutionary path so the selection pressure can be effective.

Nonetheless the genetic selection pressure still penetrates into the genome. Since agents still have early performance based on their genome, early reproductive performance is still determined by the reproductive actions in the genome. In our simulations the expected number of days before a

new agent participates in its first social learning gathering is quite small. After this point in the agent's life the selection pressure is shielded by the memome but before this point it can favor agents that have more reproductive actions in the genome.

Individual Learning Actions Learning actions in the non-learning group were null actions and were slightly maladaptive. The maladaptivity came from the fact that the action had a temporal cost of one time unit that could have been better spent on harvesting. This would suggest that there would be selection against having this trait in the genome. We see that the average memome (and thus genome) in the non-learning group has around 0.5 occurrences of each null learning action or about 1 full occurrence of a null learning action (see Figure 2). In our simulation null actions and no actions are rewarded the same in fitness.

Then the frequency of null learning actions in the memome of our non-learners speaks to the average number of time units not spent on harvesting. Specifically we'd expect an equal number of empty spaces to null learning actions (since they are equally favored by the fitness function) so an average of one null learning action suggests an average of two time units not spent on harvesting in the genome of non-learners. These unused time units are integral to the formation of more fit variants as they are replaced with new harvesting actions.

Individual learning is adaptive for the individual learning agents. We see a very rapid assimilation of the individual learning action into the population after its first appearance in the genosphere. Since with our settings it takes about 100 days to replace the entire population we expect it takes about this minimum time for the individual learning action to be represented in every living agent. That is, non-learning agents are quickly replaced by individual learning agents in the population once learning agents evolve.

This selection for individual learning continues after this initial assimilation in both the memome and the genome. For instance we see early on that the memome's average frequency of individual learning actions exceeds that of the genome's average frequency. The first agents that evolved to learn then quickly learned to learn more (their memome's acquired a second or third occurrence of the individual learning action to accelerate learning). In addition, selection for learning actions in the genome continues at a rapid rate, eventually overtaking the frequency in the memome at around two individual learning actions per agent on average.

This switch is best explained with reference to when individual learning is the most advantageous to an agent. Young agents have lower fitnesses (since their daily reward is tied to their genome) and thus to be competitive they need to rely on their individual learning actions to optimize their memome rapidly. The more rapidly they do this the smaller overall hit they will take to their fitness. Thus selection strongly

favors agents that have more individual learning actions in their genomes.

This selection does not work similarly in the memome. Older agents will have optimized their memome to a higher fitness and gain less benefit from individual learning. Indeed we see that some rare individuals learn to stop learning near the end of their lives if they can swap out that action with a more fit harvesting action. More commonly we see that agents that at birth had two or three individual learning actions in their genome will optimize at least one of those actions away by their old age.

Another interesting trend in the individual learners is the steady decrease of the of the number of individual learning actions in both the genome and the memome after peaking around 3000 days. This decrease suggests that as the simulation progress the selection pressure that once favored higher numbers of individual learning actions is lessening. This occurs as the genome optimizes and approaches the memome's fitness levels. Since the genome represents the starting conditions of the individual learning process the more fit this initial state is the less in need of learning the agents are. This happens both within the agent's life (making individual learning actions less common in memomes of older agents than younger) and within the larger population in generational time (making individual learning actions less common in the genomes of agents born later than those born earlier).

Individual learning is still an important skill in the social learning group. While social learning facilitates sharing of memeplexes and the maintenance of a cross-generational culture, it has no means of optimization. All optimization in the social learners is still due to the individual learning actions in their memomes. We see that selection for individual learning in the genomes is rapid in the social learners.

Interestingly we also see a more rapid increase in the presence of individual learning actions in the social learners' memomes in the early stages paralleling the individual learners learning to learn more. However, in the social learners this rapid increase is very quickly countered when the fitness of the shared memeplex reaches near optimal and a new pressure to learn less appears. The consequence of this is that the shared memeplex of the social learners tends to have exactly one individual learning action. Notice this is the minimum number to ensure continued optimization allowing it to satisfy the pressure for continued optimization while minimizing the cost of that optimization.

Further in rare cases the optimization is capable of eliminating that last individual learning action from the memome since there is no need for further optimization when it is at the near optimal. The reason this is a rare occurrence is because for this to occur the agent would have to use an individual learning action to optimize its genome and then eliminate the action that did the optimization and result in a genome that is more fit. This can only occur with a very lucky sequence of mutations in our setup and so is very rare.

Nonetheless this is the reason why the average number of individual learning actions in the memome is not exactly one but instead less than, indicating the number of runs in which the social learners were capable of achieving the lucky sequence of mutations to eliminate this action. About one fifth to one quarter of the simulations had eliminated this action by the end of the epoch.

Again the striking difference in shapes between the curves of the individual learning actions in the memome and genome of the social learners can be attributed to a decoupling of selection pressures on the memome and the genome. While young agents will be more likely to individually learn as soon as they socially learn their genetic proclivity is overridden by the shared memeplex's behavior.

Social Learning Actions Finally consider the social learning actions. We know these actions are null actions in both the non-learners and individual learners and as such we see very low frequencies in both the genomes and memomes of these agents. For the reasons discussed above regarding tie breaks we see non-zero occurrences of these actions.

However in the social learners the social learning action is efficacious. Further social learning is adaptive for these agents, but even more so if it is accompanied by an individual learning action. Without the individual learning action a collection of social learners will only share the most fit memeplex in their memomes. Since they are not engaged in individual learning this represents the most fit geneplex in their genome. As a result agents can still benefit from social learning in the absence of individual learning, though this benefit is negligible.

There is a parallel and independent selection pressure for individual learning. Both selection pressures are strong and so agents are expected to quickly evolve these actions, but very commonly they evolve roughly simultaneously. The two pressures are synergistic in that agents that evolve both traits outperform those with only one. The results of this synergistic selection is that the genomes of agents rapidly include at least one copy of each of the learning actions, with social learning actions being selected for more aggressively than individual learning actions.

We also see that selection for the social learning action in the genome continues to increase whereas in the memome this frequency is highly stable with an average of one per agent. Indeed the reason that this average is greater than one is that newborn agents are frequently born with more than one social learning action in their genome (and thus their initial memome) and this slight deviation from the norm of a few agents in the social community is enough to keep this average slightly over one. This is further evidence of the decoupling of the memosphere and the genosphere.

This decoupling leads to a very interesting dynamic. We see that social learners optimize to have on average about 2.5 social learning actions in the genome of an agent. A

mature adult (one that has already engaged in social learning at least once) will have only one social learning action in the memome of the agent. This means that a newborn or immature agent will be 2.5 times more likely to engage in social learning than a mature adult.

This is an important feature since this means that younger, immature social learners are more likely to engage in the activity that will make them more fit, but once they have engaged in that activity they do not attempt to perform it as frequently anymore allowing for younger agents to have the opportunity. It is also still important for mature agents to engage in social learning but their role is as an instructor, not a learner in most of these meetings.

The pressures that lead to this state are once again independent and decoupled. Young agents, due to selection on early performance on their genome, face a strong selection pressure to socialize early. This is accommodated by having as many social learning actions as possible in their genome. Old agents, due to selection pressure related to fitness, face a strong selection pressure to socialize less to make room for more harvesting. This is accommodated by having the minimum number of socializating actions as possible in the memome. The equilibrium found by this interaction is to make young agents socialize as soon as they can while older agents socializing only enough to ensure that young agents learn the dominant memeplexes in the memosphere.

Once again this is clear evidence of the decoupling of the memosphere from the genosphere in the social learners. Nonetheless, the early stages of a memome's development is still dependent on the genome of the agent. Once the agent learns from the community this dependence ends and the systems are decoupled.

Conclusions

Our experiment shows how social learning, individual learning and genetic adaptation can co-exist in a population of agents and some of the interactive dynamics between these optimizing forces. Specifically we see both instances of accelerated and decelerated genetic adaptation in response to these learning mechanisms in our model.

We see clear selection shielding in the early stages of the simulation as a result of both individual learning and social learning. The cause of this shielding is that the learning mechanisms flatten the fitness function, thus equalizing selection pressure on all agents. Later in the simulation we see slight selection boosting for the individual learners. The cause of the boosting is selection for early performance and a Baldwin effect.

In both the learning populations we see shielding of selection pressures for reproductive actions in the genome. These selection pressures still exist but are weaker in agents capable of individual and/or social learning. Shielding is greatest in social learners. This indicates that individual learning and social learning in particular can weaken or even eliminate

selection pressure for genetic reproductive behavior while simultaneously suppressing the same behaviors.

In both learning populations we also see boosting of selection pressures for learning actions in a genome. The fact that both learning mechanisms are highly adaptive leads to learners always outperforming non-learners and thus rapid assimilation of new learning actions into the genome occurs. Further we see a trend of increased selection for learning actions in the genome even when there is contrasted selection for removing those actions from the memome. This emerges from the opposite need of young and old agents for learning. The young agents are necessarily less fit and so can really benefit from learning, whereas the old agents tend to be highly optimized and thus further learning is wasted effort.

This trend appears in both individual learners and social learners but is most apparent in the social learners. This is due the evolution of the memosphere decoupling from the evolution of the genosphere made possible by the shared memosphere. Not only is this decoupling responsible for the rapid optimization of the social learning group but it results in divergent and uncorrelated behavior frequencies between genome and memome.

To be clear the evolution of the memosphere and genosphere in the individual learners is independent but the starting position of both optimization procedures begin with the genome, and thus the two procedures are coupled. In the social learners the genosphere and memosphere are coupled for a very short period before the memosphere reinitializes with the best memeplex from the community. This allows the memosphere's evolution to begin from a different point than the genosphere's evolution. This implies the decoupling is not 100% but there is still a small link. We say that while decoupled the memosphere is still tethered to the genosphere.

The decoupling of the two evolutionary spheres allows for opposite selection pressures to be fully efficacious in each sphere. The best evidence of this we have is that while selection for reproductive, individual learning, and social learning actions is strong in the genosphere of social learning agents, it is weak, or even negative, in the memosphere of social learning agents. The memosphere attempts to eliminate the extraneous reproduction actions and learning actions, while the genosphere is selecting for greater numbers of these actions in the genome. The greater numbers in the genosphere lead to the emergent behavior that young agents are more likely to reproduce, learn on their own, or learn from others than their older counterparts. This is beneficial to them since they are the least fit members of the population at this time.

References

Acerbi, A. and Parisi, D. (2006). Cultural transmission between and within generations. *Artificial Societies and Social Simulation*, 9.

- Ancel, L. (2000). Undermining the baldwin expediting effect: Does pheontypic plasticity accelerate evolution? *Theoretical Population Biology*, 58:307–319.
- Anderson, R. W. (1995). Learning and evolution: A quantitative genetic approach. *Journal of Theoretical Biology*, 175:89– 101.
- Borenstein, E., Meilijson, I., and Ruppin, E. (2006). The effect of phenotypic plasticity on evolution in multipeaked fitness landscapes. *Journal of Evolutionary Biology*, 19:1555–1570.
- Denaro, D. and Parisi, D. (1997). Cultrual evolution in a population of neural networks. In Marinaro, M. and Tagliaferri, R., editors, *Proceedings of neural nets wirn-96*, pages 100–111. Springer.
- Dopazo, H., Gordon, M. B., Perazzo, R., and Risau-Gusman, S. (2000). Undermining the baldwin expediting effect: Does pheontypic plasticity accelerate evolution? *Theoretical Population Biology*, 58:307–319.
- Fontanari, J. and Meir, R. (1990). The effect of learning on the evolution of asexual populations. *Complex Systems*, 4:401–414
- Hinton, G. E. and Nowlan, S. J. (1987). How learning can guide evolution. *Complex Systems*, 1:495–502.
- Lande, R. (2009). Adaption to an extraordinary environment by evolution of phenotypic plasticity and genetic assimilation. *Journal of Evolutionary Biology*, 22:1435–1446.
- Lindblom, J. and Ziemke, T. (2003). Social situatedness of natural and artificial intelligence: Vygotsky and beyond. *Adaptive Behavior*, 11:79–96.
- Marriott, C., Parker, J., and Denzinger, J. (2010). Imitation as a mechanism of cultural transmission. *Artificial Life*, 16:21– 37.
- Mayley, G. (1997). Guiding or hiding: Explorations into the effects of learning on the rate of evolution. In Husbands, P. and Harvey, I., editors, *Proceedings of the fourth European conference on artificial life*. MIT Press, Cambridge, MA.
- Mesoudi, A., Whiten, A., and Laland, K. (2006). Towards a unified science of cultural evolution. *Behavioral and Brain Sciences*, 29:329–383.
- Paenke, I., Kawecki, T., and Sendhoff, B. (2006). On the influence of lifetime learning on selection pressure. In Rocha, L., Yaeger, L., Bedau, M., Floreano, D., Goldstone, R., and Vespignani, A., editors, *Artificial life X*. MIT Press.
- Paenke, I., Sendhoff, B., and Kawecki, T. J. (2007). Influence of plasticity and learning on evolution under directional selection. *The American Naturalist*, 170:47–58.
- Papaj, D. R. (1994). Optimizing learning and its effect on evolutionary change in behavior. In Real, L. A., editor, *Behavioral mechanisms in evolutionary ecology*. University of Chicago Press, Chicago, IL.
- Sznajder, B., Sabelis, M. W., and Egas, M. (2012). How adaptive learning affects evolution: Reviewing theory on the baldwin effect. *Evolutionary Biology*, 39:301–310.

Self-Organizing Process and Cluster Network in the Game of Life

Yoshihiko Kayama and Yasumasa Imamura

Department of Media and Information, BAIKA Women's University 2-19-5 Shukuno-sho, Ibaraki 567-8578, Osaka, Japan y_kayama@ieee.org

Abstract

Conway's Game of Life, which is one of the most studied cellular automata, contains a self-organized rest state with residual clusters of live cells including the well-known patterns like Block and Beehive. The present article aims to discuss the self-organizing process from the viewpoint of a network representation of the Game of Life. Scale-free property of the network associated with the rest state is resilient against consecutive removals of a hub node. Of the two types of residual clusters, whose links continue to grow or not over time, the type of clusters accompanied by growing links is essential in the self-organizing process. Mixing the two types of clusters, which diversifies scales of branch graphs that correspond to avalanches caused by one-cell perturbations in the rest state, contributes to the scale-free property of the rest state. Furthermore, a network of clusters can be obtained from the rest-state network by regarding the clusters as composites of live cells.

Introduction

Conway's Game of Life, or Life, is one of the most famous cellular automata (CA) and is defined on a two-dimensional square lattice of cells, where each cell has two state values, true or false, associated with live or dead cells (Gardner, 1970; Berlekamp et al., 1982).

Around 1990, researchers debated how to verify self-organized criticality (SOC) in Life, where SOC was defined by Bak et al. (1987) in a general theory of self-organization and emergence (Bak et al., 1989; Bennett, 1991; Alstrøm and Leão, 1994; Hemmingsson, 1995; Blok and Bergersen, 1997). Subsequently, contradictory results of research in this field are thought to have originated from the different lattice sizes and/or boundary conditions adopted by the researchers. In any case, Life with periodic boundary conditions is scale-free to be regarded as "sub-critical." Thus, the main purpose of this study is to investigate the self-organizing process in Life by analyzing properties of power-law scaling or scale-free property in complex networks (Barabási and Albert, 1999).

In recent years, Kayama proposed a network representation (NR) of CA that focused not on cell values but on relationships between cells, which were related by the effects of a one-cell perturbation after a time interval. This

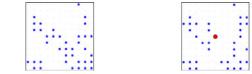
approach was used to study dynamical aspects of CA by analyzing various characteristic parameters used in network theory (Kayama, 2010, 2011, 2012). In particular, the NR of Life shows some interesting features: the well-known patterns surviving in a rest state are accompanied by characteristic networks which are divided into two types: their links continue to grow or not over time [Figures in the appendix (Kayama and Imamura, 2011)]. The rest-state network, which encompasses the patterns' networks and connects the patterns together, is a visualization of underlying tension that causes the growth of avalanches catalyzed by one-cell perturbations. The scale-free property of the reststate network was also studied by analyzing its out-degree distribution (Kayama, 2013). This article discusses the selforganizing process in Life from the viewpoint of the NR. The two types of patterns surviving in the rest state play crucial roles in the self-organizing process and the scale-free property of the rest state. From here on, we call such residual patterns "clusters" to distinguish them from all other chaotic patterns.

The next section is devoted to introducing the NR of Life by expanding the discussions of Kayama (2013). The third section explains the role of the cluster's networks in the process of self-organization. A survey of the self-organizing process is given in the fourth section where a network connecting clusters and its degree distribution are also discussed.

Scale-free Property and Its Resilience

Concrete steps to obtain a network are as follows: First, a dot pattern is obtained from a configuration after an interval time t_I [Figure 1a]. Second, another dot pattern is generated from the same configuration with a one-cell perturbation after t_I [Figure 1b]. Third, the difference of the patterns is obtained by binary addition [Figure 1c]. Fourth, a graph is obtained by connecting the initially changed cell and the changed cells after t_I [Figure 1d]. Finally, the network is defined by combining all graphs obtained from all one-cell perturbations together.

After a transient period of self-organization, any initial



(a) Dot pattern obtained from (b) Another pattern after t_I a configuration after t_I . with an initial one-cell perturbation.



(c) Changed cells obtained (d) Graph obtained by confrom binary addition of (a) and necting the perturbed cell and (b). the changed cells.

Figure 1: Concrete steps to obtain a graph (d) from a perturbed cell. A network is defined by combining such graphs obtained from all one-cell perturbations together. The central (red) cell is an initially perturbed cell and the lines are directed links, where red, blue and green indicate that the link is exiting (out-link), entering (in-link) and intermediate, respectively.

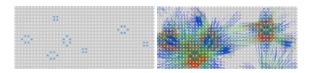


Figure 2: A rest state (left panel) and its network at $t_I=50$ (right panel). The blue and white squares in the figures represent the live and dead cells, respectively. Directed links are drawn with a color gradient from red to blue.

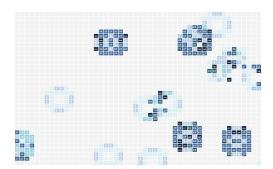


Figure 3: Out degrees in a rest state illustrated by a color gradient. Darker color means a larger out-degree.

configuration of Life reaches a rest state that contains static and oscillating patterns (clusters). Moving clusters, such as Glider, may exist if they move along closed paths when periodic boundary conditions are adopted on the lattice. Each residual cluster is separated from other clusters by enough space of dead cells that the Life rule makes no connection between them in a time evolution. Avalanches may occur because of one-cell perturbations of the rest state, and each avalanche is visualized in the NR as a branch graph originating from a perturbed cell. The rest-state network is constructed as an entire graph that encompasses all such branch graphs.

After a long enough time t_I , almost all clusters are connected to one another by long-branch graphs, which visualizes the tension that makes an avalanche grow [Figure 2]. In network theory, node degree, which is the number of links connecting a node, is one of the most popular indices used to represent an attribute of network structure. All links contained in the NR are directed because the growth over time of the corresponding avalanches is directed. In this case, the node degree can be divided into two types, out-degree and in-degree, which are the number of links departing from and entering the node, respectively. Because we focus on self-organizing processes, that is, evolution over time, we mainly discuss out-degree in the following. A branch graph is obtained by comparing two states at t_I : one in which an avalanche occurred and one in which no avalanche occurred. Because all cells with different values between states are considered to have been influenced by the one-cell perturbation that caused the avalanche, these cells are connected by directed (out) links from the perturbed cell. Thus, the out-degree of the perturbed cell corresponds to the number of influenced cells. Out-degrees are illustrated by a color gradient in Figure 3, where a darker color means a larger out-degree. Although repetitive (or stable) and chaotic CA [otherwise known as Class II and III rules according to Wolfram's classification (Wolfram, 1983)] are accompanied by localized and random networks, as expected, complex or Class IV rules do not necessarily have scale-free networks. By using out-degree as an order parameter to verify SOC together with avalanche lifetime and size, Kayama (2013) investigated power-law scaling in rest-state networks. Figure 4 shows an improved result for the distribution of avalanche scales on an N=1001 lattice, where rest-state periodicity and GPU programing were used. The time interval t_I was set to 3×10^4 , and in total, 100 initial configurations were pseudo randomly generated. Approximately 2×10^7 nontrivial avalanches were obtained from nearly 10⁸ one-cell perturbations. Table 1 gives the results of a goodness-of-fit test of power-law scaling (Clauset et al., 2009), which includes a finite-size scaling analysis of the normalized distributions of avalanche lifetimes, sizes, and out-degrees (Fisher

¹AMD Aparapi, "Aparapi - api for data parallel java." https://code.google.com/p/aparapi/

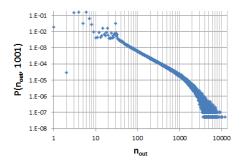


Figure 4: Normalized distribution of out-degrees of 100 reststate networks with N=1001.

Data set	<i>p</i> -value	x_{min}	Scaling	Critical
Lifetime	1.00	363	a = 1.25	$\alpha = 1.83$
Size	0.84	61	b = 1.63	$\beta = 1.86$
Out-degree	1.00	684	c = 1.2	$\gamma = 1.74$

Table 1: Results of power-law tests of avalanche scales of 100 rest-state networks with N=1001. Finite-size scaling effects were considered.

and Barber, 1972; Barber, 1983). For the distributions D(l), D(s), and $P(n_{out})$ of these indices, we postulate the following:

$$\begin{array}{cccc} D(l,\,N) & \propto & l^{-\alpha}exp(-l/l_\xi) & N,\,l \gg 1 \\ D(s,\,N) & \propto & s^{-\beta}exp(-s/s_\xi) & N,\,s \gg 1 \\ P(n_{out},\,N) & \propto & n_{out}^{-\gamma}exp(-n_{out}/n_\xi) & N,\,n_{out} \gg 1, \end{array}$$

where $l_{\xi} \propto N^a$, $s_{\xi} \propto N^b$, and $n_{\xi} \propto N^c$ are the characteristic lengths of the avalanche lifetime, size, and out-degree, respectively. The p values in the table all indicate that the indices follow a power law.

Static scale-free networks are, in general, vulnerable to attack on hub nodes. When we try to verify such an attribute in the NR, similar discussions are not applicable because the networks are dynamic and any perturbation of multi cells is forbidden within the framework of the NR. In this case, a hub node can be defined as a cell with large out-degree. An alternative way to remove multi cells is to add several onecell perturbations to a sequential series of rest states: the first rest state is reached upon adding a one-cell perturbation to an initial rest state, the second rest state is reached upon adding a one-cell perturbation to the first rest state, and so on. Because a large out-degree may become two times the number of residual cells, removing such a hub node changes the rest state into an entirely different rest state. Thus, we can expect that, even if hub nodes are subsequently removed as mentioned above, scale-free property will be preserved. Figure 5 shows a normalized distribution of out-degrees of 99^{th} rest-state network obtained from 20 initial rest states by removing a hub node 99 times from each rest state on a

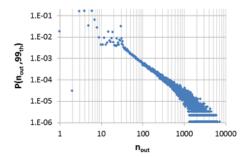


Figure 5: Normalized out-degree distribution of 20 rest-state networks after 99 times consecutive removal of a hub node on a N=500 lattice (log-log scale).

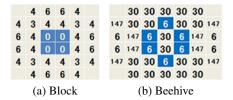


Figure 6: Out-degrees of cells surrounding isolated (a) Block and (b) Beehive clusters with N=100 and $t_I=990$. Blue cells are alive. The numbers indicate the out-degree of each cell.

N=500 lattice. The time interval between two consecutive removals was $t_I=3\times 10^4$. The goodness-of-fit test of power-law scaling of the distribution gives p=1.00. There are no remarkable changes in the distribution of intermediate rest-state networks from the first to 98^{th} rest state. As a result, the rest-state network of Life is resilient against consecutive removals of a hub node.

Cluster Network

Because there must be a causal relationship between the variety of clusters and the scale-free property in the distribution of avalanche scales, we now discuss the information the network view offers to better understand the scale-free property or self-organizing process. In a previous study, we focused on individual networks of well-known clusters and reported that two types of clusters exist: those whose links continue to grow or not over time (Kayama and Imamura, 2011). Herein we call these types as type-G for those that grow and type-NG for those that do not grow. Networks of some well-known clusters are listed in the appendix. The most frequently encountered type-G and type-NG clusters are the Beehive and Block clusters, respectively. Out-degree distributions of their networks are illustrated in Figure 6, where the numbers indicate the out-degree of each cell. When a rest state is made only from Block clusters sufficiently separated (greater than four cells between them) so

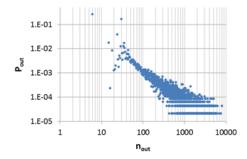


Figure 7: Normalized out-degree distribution of a rest-state network made from 1250 Beehives with $t_I = 2 \times 10^4$ and N = 500.

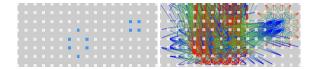


Figure 8: Block cluster terminates Beehive links.

that there is no interference, the out-degree distribution is apparently not scale free but has peaks only at $n_{out} = 3$, 4, and 6. However, if a rest state is made only from randomly located Beehive clusters, its out-degree distribution is scale free; see Figure 7, for which $t_I = 2 \times 10^4$ and N=500, where the number of Beehive clusters was determined to be 1250 from the average number of residual live cells (approximately 3% of total cells). Following the same process of making a rest state from only type-G clusters, we confirm that the rest state is also accompanied by a scale-free network. These results mean that type-G clusters are essential elements for the scale-free property². When Beehive and Block clusters are mixed to make a rest state, a scale-free distribution is also obtained. However, the scaling exponent is larger than for a rest state made only from Beehive clusters, which means Block clusters act to restrain Beehive links from growing. A visual confirmation of this effect is shown in Figure 8, where Beehive links are terminated by a Block cluster.

What happens if a hub node in the state of only one Beehive cluster in Figure 6b is removed? The rest state changed by the avalanche contains many kinds of clusters and the out-degree distribution of its network is like a scale-free distribution. Since even one Beehive cluster can induce a variety of clusters by the removal of a hub node, the resilience of scale-free property discussed in the previous section is a natural consequence.

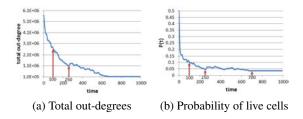


Figure 9: (a) Total out-degrees and (b) probability of live cells as a function of time with N=100. Vertical arrows indicate the times when out-degree distributions are shown in Figure 10.

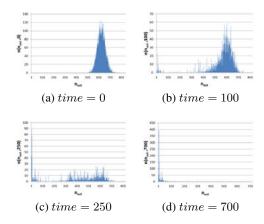


Figure 10: Out-degree distributions of state networks in a self-organizing process from a pseudo randomly generated initial configuration at $time=0,\,100,\,250,\,$ and 700 with N=100 and $t_I=2\times 10^4-time.$

Self-Organizing Process and Network of Clusters

We now illustrate the transition process from a random initial configuration to a rest state. Figure 9 shows the total outdegrees and probability of live cells as a function of time. Figure 10 shows out-degree distributions of state networks at $time=0,\,100,\,250,\,$ and 700. The state at time=700 has already reached the rest sate.

- 1. Because the initial network is composed of a lot of random long-range links, its out-degree distribution is almost normal [Figure 10a].
- 2. After several time steps, the number of live cells rapidly declines. Some Block clusters are already synthesized and restrain long-range networks. In addition to losing the shape of the normal distribution, the number of short-range links increases [Figure 10b].
- 3. Chaotic regions settle into cells that are almost dead except for the clusters. The peak of the normal distribution

²The interference of closely spaced type-NG clusters may act as Type-G clusters.

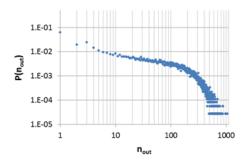


Figure 11: Normalized out-degree distribution of 10 cluster networks with N=1000.

almost collapses and large-scale branch graphs of type-G clusters remain [Figure 10c].

4. Cluster-synthesizing processes continue until the fraction of live cells reaches approximately 3%. Next, tiny unstable regions or some moving clusters, such as Glider, move and collide with other clusters, causing localized chaotic interactions. Through such unstable states, the configuration reaches a rest state with a scale-free network [Figure 10d].

When clusters are regarded as composite states of live cells, it is interesting to study the network connecting them. The NR approach makes it possible to visualize such networks of clusters. In Figures 6a or 6b, we define a cluster group as the cells that constitute a cluster plus the surrounding dead cells with non zero out-degree. If a one-cell perturbation of the group of cells influences other clusters, these clusters are considered to be linked with the perturbed cluster group. Figure 11 shows a normalized out-degree distribution of cluster networks obtained from ten different rest states with N=1000. The average scale of cluster groups (larger than a 10×10 rectangle) is too large to determine if the distribution follows a power-law. Further investigation would require using a larger lattice size.

Conclusions

In this article we studied the self-organizing process of Life through the NR approach, which, unlike dot patterns, can reveal dynamical aspects because it contains a history of changes of cell states. In particular, associated networks of clusters in a rest state can be classified into growth type and non-growth type (types G and NG, respectively) according to the growth of their network. By analyzing the scale-free property of the rest-state network, we noticed that type-G clusters are essential in the self-organizing process to connect clusters through long-range links. Type-NG clusters also play the important role of diversifying scales of branch graphs to restrain the growth of type-G links. The NR of Life also shows that it is possible to visualize a cluster net-

work when each cluster is recognized as a composite state of live cells.

Although the ratio in which different cluster types are mixed must play an important role in the self-organizing process, no clear discussion or demonstration of this effects is presently possible. Further investigation into this ratio and cluster networks requires calculations on a larger lattice and analytical derivations.

References

- Alstrøm, P. and Leão, J. (1994). Self-organized criticality in the "game of life". *Phys. Rev. E*, 49:R2507–R2508.
- Bak, P., Chen, K., and Creutz, M. (1989). Self-organized criticality in the 'game of life'. *Nature (London)*, 342:780.
- Bak, P., Tang, C., and Wiesenfeld, K. (1987). Self-organized criticality: an explanation of 1/f noise. *Physical Review Letters*, 59 (4):381–384.
- Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286:509–512.
- Barber, M. N. (1983). Finite size scaling, in Phase Transitions and critical phenomena, volume 8. Academic Press, London.
- Bennett, C. (1991). 'life'not critical?. Nature, 350:468.
- Berlekamp, E. R., Conway, J. H., and Guy, R. K. (1982). *Winning Ways for Your Mathematical Plays*. Academic, New York.
- Blok, H. J. and Bergersen, B. (1997). Effect of boundary conditions on scaling in the game of life. *Physical Review E*, 55.5:6249–6252.
- Clauset, A., Shalizi, C. R., and Newman, M. E. J. (2009). Power-law distributions in empirical data. *SIAM Review*, 51:661–703.
- Fisher, M. E. and Barber, M. N. (1972). Scaling theory for finite-size effects in the critical region. *Phys. Rev. Lett.*, 28:1516–1519.
- Gardner, M. (1970). Mathematical games. *Scientific American*, 223:102–123.
- Hemmingsson, J. (1995). Consistent results on ¡Ælife¡Ç. *Physica D: Nonlinear Phenomena*, 80.1:151–153.
- Kayama, Y. (2010). Complex networks derived from cellular automata. arXiv:1009.4509.
- Kayama, Y. (2011). Network representation of cellular automata. In 2011 IEEE Symposium on Artificial Life (IEEE ALIFE 2011) at SSCI 2011, pages 194–202.

Kayama, Y. (2012). Network view of binary cellular automata. In Sirakoulis, G. and Bandini, S., editors, Cellular Automata, volume 7495 of Lecture Notes in Computer Science, pages 224–233. Springer Berlin / Heidelberg. 10.1007/978-3-642-33350-7_23.

Kayama, Y. (2013). Network representation of the game of life and self-organized criticality. In *Artificial Life (AL-IFE)*, 2013 IEEE Symposium on, pages 60–66. IEEE.

Kayama, Y. and Imamura, Y. (2011). Network representation of the game of life. *Journal of Artificial Intelligence and Soft Computing Research*, 1 (3):233–240.

Wolfram, S. (1983). Statistical mechanics of cellular automata. *Rev. Mod. Phys.*, 55:601–644.

Appendix: Networks of Well-known Clusters

We present some network examples of well-known clusters in Life, including the simplest static clusters, "still lifes"; repeating clusters, "oscillators"; and a moving cluster, "spaceship" (Kayama and Imamura, 2011). In the figures, the blue and the white squares represent live and dead cells, respectively. Directed links are drawn by a color gradient from red to blue. Red denotes that the link is departing from the node (out-edge), and blue denotes that the link is entering the node (in-edge).

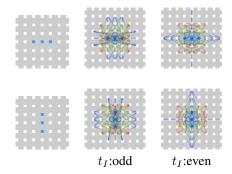


Figure 14: Blinker cluster (type-NG) and its oscillating networks (period 2), which depend on t_I parity.

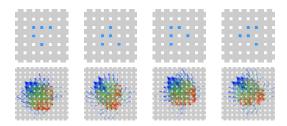


Figure 15: Spaceship cluster (Glider: type-G) and its moving networks (period 4) at $t_I = 12$.

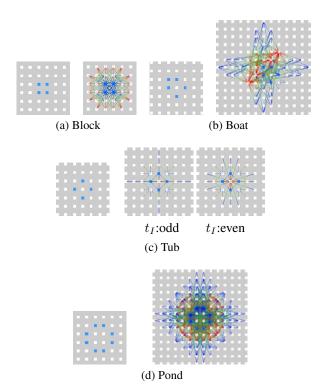


Figure 12: Still lifes (type-NG) and their networks which stop growing at some t_I value. (c) Tub network depends on t_I parity.

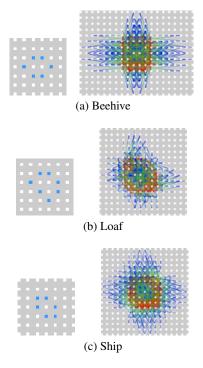


Figure 13: Still lifes (type-G) and their networks which grow with t_I . Networks were obtained at $t_I = 20$.

Adaptive Division of Labor in Multi-Robot System with Minimum Task Switching

Wonki Lee¹ and DaeEun Kim¹

¹Biological Cybernetics Lab, School of Electrical and Electronic Engineering, Yonsei University, 50 Yonsei-ro, Seodaemun-gu, Seoul, 120-749, South Korea wonkilee@yonsei.ac.kr

Abstract

In many multi-robot systems, various tasks are allocated dynamically to an individual robot and each robot should decide its own work that is the best commensurate with its current state. To solve complex task allocation problems, agentbased approaches based on the model of division of labor of many social insects have gained increasing attentions in recent years. In this paper, we consider the problem of adjusting the ratio of robots equally to the ratio of given tasks to handle the division of labor dynamically with less number of task switches. Inspired by several insect societies displaying an effective division of labor with the limited abilities, the response threshold model is applied. An Individual robot has a limited, constant-sized task queue and the information obtained from the observation behavior is stored within this queue. Using the ratio of tasks in queue and the predefined response threshold values for all possible tasks, an individual agent decide its task and to handle the desired division of labor dynamically and obtains the specialization for the specific tasks that induces the less number of task switching. To show the robustness and flexibility of our proposed method, various experiments are executed and the results are compared with an other method.

Introduction

Generally speaking, working together can obtain better performance to complete a given task than working alone. Therefore, the concept of swarm intelligence is important to design a cooperation model in multi-agents systems. In multi-agent systems, there are two issues to perform tasks. One is cooperation and the other is division of labor. The focus of cooperation is to achieve a difficult task that a single robot cannot solve alone, and the focus of division of labor is to manage labors efficiently which take lots of times and costs in performing. One approach to these kinds of problems is to establish an appropriate scheduling as an adaptive process that adapts the number of agents that perform the same task in a dynamically changing environment. There are many examples of effective, adaptive behaviors in multiagent systems and the swarm intelligence has been served as inspiration for multi-agent optimization and control algorithms in recent years.

All individuals in the swarm systems have the same and simple sets of behavioral rules and perform the same activities. However, several insect societies with the limited abilities display an effective division of labor using specialization that different activities are simultaneously performed by specialized individuals (Robinson, 1992). In order to explain these results, the response threshold model where an individual performs a task if the stimulus associated with given task exceeds its threshold is proposed (Theraulaz et al., 1998). An individual with a low response threshold value will tend to perform the corresponding task even if the stimulus is very low. Conversely, an individual with a high response threshold value will perform the corresponding task only if the stimulus is very high. This response threshold model can explain the regulation system of a product, such as food, by allocating individuals to forage as soon as the food demand increases. Examples of division of labor include foraging and nest defense in ants (Detrain and Pasteels, 1991), foraging and nursing (Calderone and PAGE Jr, 1996), and nectar and pollen collection (Visscher and Dukas, 1995) in honeybees.

Individuals can be specialized in a strong or soft manner. Strongly specialized individual only perform one or a few activities. Although genetic factors certainly play a role in some types of strong specialization (Sundstrom, 1995), such as polyethism (age-dependent specialization) and polymorphism (different body shapes), the unpredictable modifications of the environment and the variability within the colony require an additional mechanism to ensuring dynamic task allocation for colony survival. So, the softly specialized individual performs several activities, but tends to perform the activity that is the most needed by the group at every instant. For example, some specialists of type A can carry out task B that they would normally not perform, if the number of specialist of type B decreases (Wilson, 1984).

Task allocations in social insects have been extensively studied since their divisions of labor are realized by simple rules. The fixed response threshold model is an important theory to explain the regulation of division of labor in insect societies based on fixed response threshold value (Theraulaz

et al., 1998). This model generates the different response depending on the intensity of a stimulus and the response threshold value and determines the tendency of an individual to a particular task. Task allocation problem based on the fixed response threshold model was studied to deal with labor regulation (Yang et al., 2009). Furthermore, an improved version that dynamically adapts threshold values has been studied. A computational model of how wasp colonies coordinate individual activities and allocate tasks to meet the collective needs is proposed in study of (Cicirello and Smith, 2004). In the study of (Lichocki et al., 2012), the response threshold model is represented by the artificial neural networks and provides a mechanism to regulate division of labor in social insects. However, the method such as neural net or genetic algorithm is not suitable in a distributed system and if we approach in a centralized system, all information is shared among the robots and an individual robot knows the ratio of entire tasks, then robots can choose its own behaviors easily as the fraction of tasks. In our work, the entire information is not available to an individual robot and it is nature to use the local knowledge alternatively to achieve the effective division of labor.

In this paper, we consider the problem of dynamically adjusting the ratio of robots equally to the ratio of given tasks with less number of task switching. Task switching needs a non-productive period of time which is used to be ready for processing the different type of current task and may need additional costs in many factory applications, so it is recommended to reduce task changes for improving the performance of system. In the following sections we will describe our task and simulation results display the robustness and adaptability to environmental variations of our proposed model.

Task Description

The important part of swarm system is surely how to measure the overall performance of multi-robot system. In this paper, the way of checking the performance of system is used by foraging task as shown in the paper of (Jones and Mataric, 2003). One of two kind of colors is assigned to robots and each robot performs a task to forage the closest puck whose color is same with the its current assigned color. Simultaneously, robots perform avoiding obstacles for preventing collision with other robots and barrier of a circular arena, and change its current color if necessary. Especially, this task is seen as a feeding behavior of animal swarm if robots and pucks are regarded as predators and prey, and the performance is determined by counting the number of foraged pucks and the number of task changes among all robots. In many studies, the foraging tasks are commonly selected to search for a specific object and move it properly to any storage place such as home or nest. However, robot generally spends lots time for wandering in that case. Thus, the task is set to detecting pucks and not to move them to

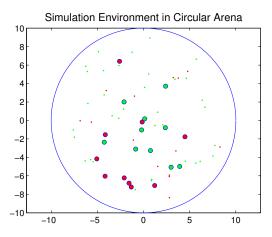


Figure 1: Snapshot of simulation environment. Robots and pucks are distributed randomly in a circular arena. The dark (red in web site) and light (green in web site) gray circles are robots that move randomly in an arena and the small dots are pucks to be foraged by robots.

handle the division of labor more accurately and quickly.

Simulation Environment Setting

Foraging task is performed in a circular arena and the simulation environment is shown in Fig. 1. The area of arena is about $315m^2$ and the radius is about 10.01m. Initially, robots and pucks are distributed randomly in an given arena and assigned to red or green color in a specific ratio. As time goes on, the robots move randomly and if the robots detect and grasp the pucks, new pucks which are the same color of foraged pucks are generated in arbitrary places. These relocations mean maintenance of the number and ratio of pucks that there are dynamically additional works from viewpoint on foraging task.

Robot Behaviors

Each robot notices surrounding situations through its sensors and decide the task based on the local information. The robot finds the closest puck whose color is same with the assigned color and turns toward the direction of the selected puck location and moves to grasp the puck. The robot behaviors are composed of camera detection, wandering, obstacle avoidance, grasping, and color switching. The details of the behaviors are explained as below.

Camera Detection Each individual robot has a camera which shoots the front of 60° with 5m range and obtains the information about how many and how far red and green pucks are located using visual information. The robot stores the obtained information in its task queue and estimates the desired ratio of puck colors by counting the number of puck colors in the queue at the color transition stage. Because the

continuous scanning shows duplicated results and produces poor estimations of the desired ratio of tasks, the robot's camera shoots at every 2m movement.

Wandering We used circular robots which is 30cm in diameter. The maximum forward/backward speed of robot is set to 0.25m per each simulation time step and robot can rotate at the same spot. Robot moves after the preceding camera detection is finished, but does not carry out wandering behavior if obstacle avoidance or grasping puck is performed.

Obstacle Avoidance Obstacle avoidance is the basic and important behavior of swarm robots to remove the collision accident with other robots or obstacles. The barrier of arena and other robots are defined as obstacles, but the pucks are not regarded as obstacle. If the obstacle avoidance is performed frequently, the robot spends lots of time on avoiding behavior and wandering or grasping behavior is not carried out in that time step, therefore accidents between robots and pucks are ignored.

Each robot has 8 IR sensors and sensors are equipped on the border and distributed uniformly so as to cover 180° on the front side of robot. The range of sensor is 1m and robot changes moving direction when any objects are detected within this sensing range. A robot turns right when the left part of sensors detect obstacles, and the robot reversely turns left when the right part of sensors detect obstacles. The amount of turning angle is set to 45° in these two cases. If both parts of sensors detect obstacles at the same time, the robot heads toward the counter direction by turning half rotation. So, the basic turning angles are set to 45° and 180°. However, if the amount of change is same, the changing pattern is same and there is a danger of congested traffic when the robots are surrounded in a specific area. Thus, we add the difference in angle with Gaussian random variables for avoiding these problems.

Grasping Robot finds the closest puck of same color and adjusts the direction toward the closest puck. If the distance between the robot and the selected puck is lower than 0.5m and there is no occurrence of obstacle avoidance, the robot grasps the puck. After grasping puck, robot moves to other place to forage next pucks.

Color Switching Each robot has to decide what color of puck it forages. Each individual robot has a limited, constant-sized task queue and the local information obtained from the observation behavior is stored within this queue. Each individual estimates the ratio of tasks in its queue and updates task switching function using an estimated ratio of tasks and the response threshold values for all possible tasks. Robot can change its current task probabilistically based on the score value of task switching function and performs the task that has the maximum score value. Through these re-

peated behaviors, each robot becomes to have a tendency to process one specific tasks, and this specialization reduces the total number of task switching with maintaining the desired global division of labor.

We propose the following task switching function based on the estimated ratio of puck colors stored in task queue and the predefined response threshold values for tasks.

$$P_{\theta_{ij}}(t) = \frac{S_{ij}(t)}{\theta_{ij}} \tag{1}$$

Eq. (1) indicates the score value for ith robot about jth color at time t and is determined by the estimated ratio $S_{ij}(t)$ and the response threshold value θ_{ij} . As the estimated ratio is measured higher or the threshold value is lower, this score value is computed higher, and robot changes the current color to the color that has the maximum score value.

If the length of the queue is increased, the estimated results will be more accurate and the change rate of the stimulus becomes smoother. Then, the number of task switching will be decreased. Considering these effects of queue length, two formulations for estimating the ratio of puck colors are proposed. The first method (called 'History1') is based on the moving average of the information deposited by the queue, and the second method (called 'History2') is based on the moving average of the value obtained from the first method. In the first approach, the estimated puck ratio $S_{ij}(t)$ is obtained by the following equation:

$$S_{ij}(t) = \frac{1}{N} \sum_{l=1}^{N} color_{ij}^{\ l}(t)$$
 (2)

with N is the length of task queue and $color_{ij}^{\ l}(t)$ is 1 if the puck color in the lth queue of robot i is color j at time t, otherwise 0. In the second approach, the stimulus $S_{ij}(t)$ is obtained by following equation:

$$S_{ij}(t) = \frac{1}{N} \sum_{k=0}^{N-1} S_{ij}(t-k) = \frac{1}{N} \frac{1}{N} \sum_{k=0}^{N-1} \sum_{l=1}^{N} color_{ij}^{l}(t-k)$$
(3)

This second method has an effect to have a NxN length of the queue and N is set to 20 in this paper.

On the other hand, every robot initially has randomly distributed fixed response threshold values for two tasks of foraging red and green color pucks. Each threshold value has a range from 0 to 1 and the sum of two values is set to 1. These fixed response threshold values are enough to handle the division of labor. Through these variations of response threshold values, the results of Eq. (1) are different about the same estimated ratio of puck colors and this specific choosing phenomenon is enough to be called as specialization about the particular color. This phenomenon affects the performance of system by reducing task changes. Therefore, the division of labor considering specialization is shown well in this

simple model. Closest to our work, the ratio model studied two different kinds of foraging task switching functions, such as simple step and probabilistic transition function using ratio of pucks and foraging robots (Jones and Mataric, 2003). However, our model differs in using a task switching function based on the response threshold model.

Experimental Results

We evaluate our model by counting the total number of foraged pucks and the number of task changes in all robots. There are 50 pucks distributed and 20 robots move around to find pucks. Initially, the first half of robots are assigned to red color to forage red color pucks and the others robots are assigned to green color to forage green color pucks. The ratio of pucks is changed during simulation to check the robustness and adaptability to the environmental variations. During the first 1000 simulation time steps, the ratio of red and green color pucks is set to 30% vs 70%, respectively. 15 red pucks and 35 green pucks are randomly distributed

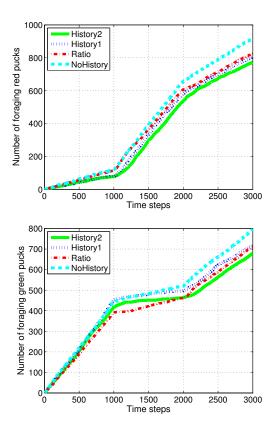


Figure 2: The total number of foraging red (top) and green (bottom) color pucks. The dash-dot line (red in web site) is the result of the ratio model, the dotted line (blue in web site) is the result of 'History1', and the solid (green in web site) line is the result of 'History2'. The dashed (cyan in web site) line is the result of 'History1' with no task queue.

in a given arena. After the first simulation period, the ratio of red and green pucks is changed to 80% vs 20%, and 50% vs 50% at every 1000 simulation time steps passing. The performances of 20 independent runs are averaged and the different positioning of robots and pucks and different values of response threshold are all selected randomly at every trial.

Fig. 2 and Fig. 3 show the results of foraging task simulations using different task allocation algorithms. The dashdot line represents the result of ratio model, the dotted line

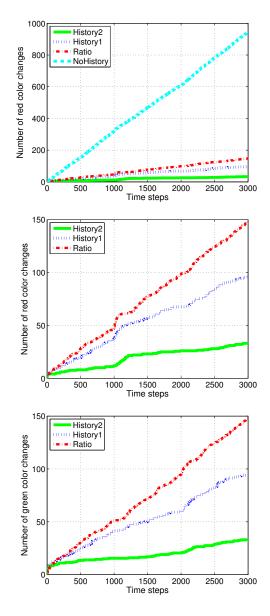


Figure 3: The total number of task changes for red color in all models (top). The number of task changes for red (middle) and green (bottom) color in three models that use task queue.

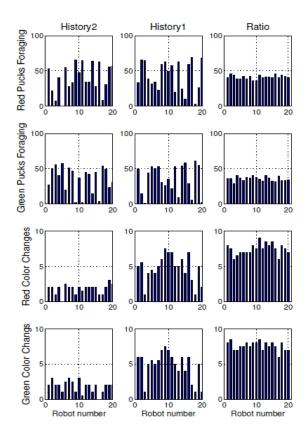


Figure 4: The detail performance of individual robot. The first column is the results of 'History2'. The figures in the second and third columns are results of 'History1' and ratio model. There are number of foraging red and green color pucks, color changes for red and green color of all robots from top to bottom direction, respectively.

is 'History1', the solid line is 'History2', and the dashed line is the result of 'History1' with no task queue. The red and green pucks are foraged steadily with the similar rate in all methods, and the number of color changes of all robots are reduced to 65.2% of red colors and 63.5% of green colors in the results of 'History1', and 22.3% of red colors and green colors in 'History2' comparing to the results of ratio model. The model that doesn't use the task queue shows many times larger of color changes than other models that use the task queue.

Our model based on the response threshold model, especially 'History2' shows the significant performances in reducing the task changes, and the specialization characteristics of robots is the most important attributes in these results. If robot starts to forage one specific color intensively, the robot is designed to have a tendency to forage the same color puck, even if the same amount of stimulus intensity is induced by each color. The specialization for an individual robot and the performances of each robot can be compared

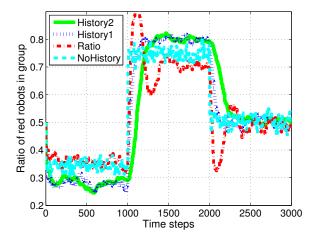


Figure 5: The overlapped trends of ratio of red color robots in group. In foraging task, the ratio of red color pucks is set to 30% initially and the ratio is changed to 80% and 50% after every 1000 simulation time steps.

in Fig. 4. The most striking feature is a task specialization. All robots in the ratio model forage red and green pucks equally, but the pucks are foraged selectively in our models. Relying on the difference of the response threshold values for two colors, some robots are strongly specialized in foraging a specific color puck. For example, in the result of 'History2', the robot 9, 11 and 16 foraged almost red color pucks and the robot 3, 5, and 17 foraged almost green color pucks, so there are few color change occurrences among these robots. However, other robots that are softly specialized forage both color pucks and the major of color changes occur among softly specialized robots.

In general, the shorter length of task queue results the fast convergence to the desired division of labor but leads to more frequent task switching for an individual agent. The Fig. 5 shows the ratio of red color robots in group during the entire simulation. From the results, we can see that 'History1' shows the faster convergence to the desired ratio than 'History2'. However, there are some differences in the results of the ratio model. The ratio model shows the fastest response to changes of environment, but it shows the worst overshooting and errors to the desired ratio of the red color robots. The ratio model focuses on matching between the estimated ratio of puck and the color ratio of robots. So, even in an extreme case, the ratio model sets a few robots to foraging the minor color pucks. It can be seen as the waste of robot resources because it is better to foraging the major portion of color pucks, but this leads to better results in foraging the minor portion of color pucks in a circular arena. This trade-off will be discussed in the next section. On the other hand, 'History1' with no task queue converges immediately and slightly off of the desired division of labor.

Discussion

The multi-robot systems can be applied to various tasks and the performance of system is not easy to be defined in multicriterion objectives. There are many variables considered in foraging tasks. For example, the results can be evaluated as the wasted time for completing tasks, consumed energy, actual amount in a division of labor, the number of foraging pucks, and the number of task switching.

If we consider the total wasted energy as the performance of system, an individual robot can spend energies for camera detection, grasping a puck, wandering distance and color change. The number of camera detections and grasping pucks are proportional to the simulation times and the wasted energies for those behaviors are not much different for all methods. In addition, the total number of wandering distance is also same as shown in Fig. 6, then the total consumed energies depend mainly on the number of task changes. So, the difference performance between the response threshold model and the ratio model will stand out if there needs some extra cost in a task switching. As the occurrences of task switching are frequently, the totally

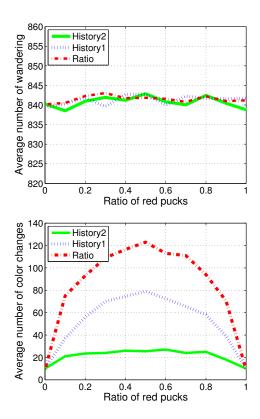


Figure 6: The number of wandering (top) and color changes (bottom) according to the ratio of red color pucks from 0% to 100% during 1000 simulation time steps. The number of wandering is almost same, but the number of color changes differs according to methods and ratio of red color pucks.

consumed energies for foraging pucks are increased more rapidly in the ratio model and the response threshold model is more appropriate when the transition cost is high.

The performance will be also changed if the ratio of puck colors is different. If the ratio of one specific color is high, the probability that the robot meets pucks of the same color is also high. It means that the number of task switching becomes smaller in that case and this leads to the improvement of result. Based on results in Fig. 6 which shows the number of color changes according the ratio of one specific color pucks, it is easy to think that both ratio model and response threshold model will show improvements as the ratio of major portion puck is increased.

However, the system is difficult to be always good at all cases when it shows high performances in some aspects. If we concern the ratio of foraging pucks as the performance of forging task, our models show the worse performance. This is because the distribution of pucks in a circular arena makes the difficulty for robots to search specific color pucks and produces difference between the ratio of red robots in group and the ratio of foraged red pucks. In the response threshold model, the robot becomes to have a specialization and tends to forage one specific color puck. So, the moving distance without task changes of robots become longer and the probability that robots meet the wanted color pucks is decreased as the square of ratio in a circular area. Especially, if the portion of two color pucks is much different, the minor color pucks have a little chance to be foraged by robots.

Fig. 7 shows the significant difference in the results of the number of foraged pucks during the first 1000 simulation time steps in Fig. 5 when the ratio of red color pucks is set to 30%. The number of foraged green color pucks is larger and the number of foraged red color pucks is smaller in the response threshold models than that of the ratio model. Theoretically, the probability that robots can find specific color pucks in a circular arena is proportional to the square of ratio of each task. When the ratio of red and green color pucks is 30% vs 70%, the probability that robots meet red color pucks is 9% / (9% + 49%). If there are no obstacle avoiding behaviors, then the theoretical ratio of foraging red puck is 15.5%. In Fig. 7 'History1' and 'History2' tend to converge slightly off of the expected ratio.

Conclusions

The division of labor generally needs the entire information of environment as an important variable. However, in many multi-robot systems, robots don't have enough knowledge about environment and each robot should use its local observation knowledge alternatively and choose the best commensurate work with its current state. In this work, we applied the response threshold model to foraging task and obtained a division of labor for performing a given foraging task more effectively. Our proposed model based on the estimated task ratio as an important variable and randomly

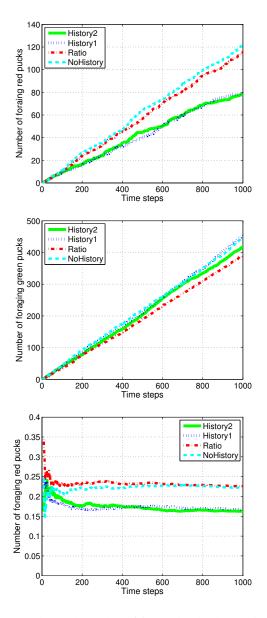


Figure 7: The total number of foraged red color pucks (top), green color pucks (middle) and the ratio of foraged red color pucks during the first 1000 simulation time steps in Fig. 5 when the ratio of red color pucks is set to 30%.

selected response threshold values for all tasks displays robust and optimal results. Fixed threshold values are enough to have a specialization tendency of robots and the overall performance of system has an advantage in minimizing the occurrences of task switching. If there needs extra costs for task switching, our method will shows an improved energy efficiency. However, we observe that there is a little trade-off between the number of foraged pucks and color changes in a foraging task using a circular arena. We will apply to various problems to check the robustness and flexibility of

our method. Moreover, there are many other variables to be measured as the performance of given task and we leave the works as a future work. Furthermore, we will study automatic methods for selecting the optimal number of groups having the same response threshold values or the optimal response threshold values for each group to reduce the task changes with maintaining the ability of division of labor.

Acknowledgements

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST) (No. 2012R1A2A4A01005677).

References

- Calderone, N. W. and PAGE Jr, R. E. (1996). Temporal polyethism and behavioural canalization in the honey bee, *Apis mellifera*. *Animal behaviour*, 51(3):631–643.
- Cicirello, V. A. and Smith, S. F. (2004). Wasp-like agents for distributed factory coordination. *Autonomous Agents and Multiagent systems*, 8(3):237–266.
- Detrain, C. and Pasteels, J. (1991). Caste differences in behavioral thresholds as a basis for polyethism during food recruitment in the ant, pheidole pallidula (nyl.)(hymenoptera: Myrmicinae). *Journal of insect behavior*, 4(2):157–176.
- Jones, C. and Mataric, M. J. (2003). Adaptive division of labor in large-scale minimalist multi-robot systems. In *Intelligent Robots and Systems*, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, volume 2, pages 1969–1974. IEEE.
- Lichocki, P., Tarapore, D., Keller, L., and Floreano, D. (2012). Neural networks as mechanisms to regulate division of labor. *The American Naturalist*, 179(3):391–400.
- Robinson, A. (1992). Dietary supplements for reproductive conditioning of crassostrea gigas kumamoto (thunberg). i. effects on gonadal development, quality of ova and larvae through metamorphosis. *Journal of Shellfish Research*, 11:437–437.
- Sundstrom, L. (1995). Sex allocation and colony maintenance in monogyne and polygyne colonies of formica truncorum (hymenoptera: Formicidae): the impact of kinship and mating structure. American Naturalist, pages 182–201.
- Theraulaz, G., Bonabeau, E., and Denuebourg, J. (1998). Response threshold reinforcements and division of labour in insect societies. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 265(1393):327–332.
- Visscher, K. P. and Dukas, R. (1995). Honey bees recognize development of nestmates' ovaries. *Animal Behaviour*, 49(2):542–544.
- Wilson, E. O. (1984). The relation between caste ratios and division of labor in the ant genus pheidole (hymenoptera: Formicidae). *Behavioral Ecology and Sociobiology*, 16(1):89–98.
- Yang, Y., Zhou, C., and Tian, Y. (2009). Swarm robots task allocation based on response threshold model. In *Autonomous Robots and Agents*, 2009. ICARA 2009. 4th International Conference on, pages 171–176. IEEE.

Mutation accumulation in bacteria exposed to UV radiation

Atsushi Shibai¹, Saburo Tsuru¹, Bei-Wen Ying¹, Daisuke Motooka², Kazuyoshi Gotoh², Shota Nakamura² and Tetsuya Yomo¹

¹ Graduate School of Information Science and Technology, Osaka University, Yamadaoka 1-5, Suita, Osaka, Japan
² Research Institute for Microbial Diseases, Osaka University, Yamadaoka 1-5, Suita, Osaka, Japan
yomo@ist.osaka-u.ac.jp

Abstract

Reducing native complexity from living organisms has significance in several aspects such as academic application as model organism and industrial application as factory of useful material. In particular, reduced complexity is also interesting in the field investigating the minimal form of "life-as-we-knowit." However, subtracting or inactivating genes from the genome without growth defects is difficult due to the complexity of gene network and error proofing functions. In this study, using model bacteria *Escherichia coli (E. coli)*, we designed a culture system using ultraviolet as a mutagen in order to select possible mutants growing rapidly with genomic inactivation. Here, we demonstrated that the culture system could accumulation of many mutations and preservation of growth ability. These results suggest that our method is effective to obtain functionally reduced genome of *E. coli*.

Introduction

E. coli is broadly used as a representative model microorganism in many biological fields such as genetics and genetic engineering. In some growing industrial fields such as pharmacy and energy production, E. coli or other microorganisms have been used as factories of useful chemical compounds. These microorganisms have intricate gene networks acquired through their evolutionary processes. Such inherent complexity often poses an obstacle to arbitrary genetic operation. Therefore, a reduction of such complexity through genomic inactivation would be helpful in several applications both academic and industrial. In addition, the attempt of functional reduction of the microorganism's genome itself has importance in investigation of the minimal form of "life-as-we-know-it."

An *E. coli* stain with 20% reduced genome has been engineered (Pósfai, et al. 2006), with the manual approach focusing on the genes which are considered as non-essential and attempting to remove them from the genome. But this approach is difficult to perform repeatedly because it is required to choose the set of genes that does not reduce the growth rate in every single operation of gene replacement. On the other hand, in nature, large genome reduction with gene inactivation is observed in endosymbionts (Mira, et al. 2001). This phenomenon is considered as the result of mutation accumulations under weak selection pressure due to the compensatory support from their hosts for the lost functions of the endosymbionts. Most mutations extinguish the function

of the gene and are neutral or deleterious for the bacterial growth (Eyre-Walker and Keightley, 2007). In order to apply that in-nature genome reduction process to evolutionary engineering, necessity of long term caused by low mutation rate and degradation of bacterial growth need to be resolved.

In this research, ultraviolet (UV) exposure was used as the mutagen which raises mutation rate as the dosage increases, and the subculture experiment was achieved with growth selection to maintain bacterial growth ability. In that way, the problems of long term and degradation of growth are supposed to be solved. This novel method allow high speed mutation fixation maintaining growth rate aiming at reduction of bacterial genome in term of the number of its functions.

Materials and Methods

We performed daily transfers of the bacterial cultures for 28 days using *E. coli* MDS42 strain whose genome was manually reduced from the progenitor. The bacteria were incubated in M63 minimal medium. Standard bactericidal lamps were used as the UV source. The UV dosages were measured and arranged by changing exposure time, distance from the lamp and usage of filters. The experiment was replicated six times and achieved simultaneously.

At the first day of the subculture experiment, a saturated pre-culture of MDS42 was diluted to 100 times with fresh M63 medium and applied 100 μ l each into 5 wells on the 96-well microplate. These 5 wells were irradiated with different dosages of UV respectively. After sealing, the plates were incubated at 37 $\,^{\circ}$ C with shaking at 550 rpm for 24 h.

As the daily operation of the subculture (Fig.1), at first, optical density (OD) of 595 nm were measured for the five wells. The wells whose UV dosage was the largest among the survived wells (OD>0.1) were sampled and diluted to 100 times with fresh M63 medium, and poured 100 μ l each into 5 wells on a new 96-well microplate.

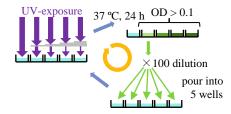


Figure 1: Cycle of the culture system with UV-exposure.

The coordination of the UV dosage was conducted with the filters, which had each a different numbers, and all wells were exposed to UV simultaneously. One filter dims UV to 0.6 times, and each well has 0, 2, 4, 6 or 8 filters, respectively. The exposure was raised as the linages obtained UV-resistance and the number of filter of the selected wells decreased. The dosage on the well without filter was tracked on a daily basis and defined as the basic dosage. The basic dosage values of 6 lineages were identical for each day through the experimentation.

The -80 °C frozen stocks of the ancestor and 6 evolved lineages (28th day) were analyzed as follows. The maximum growth rates in the absence of UV exposure were measured from the increase rate of OD values in exponential growth phase. The purified genomic DNA samples of the lineages were sequenced by Illumina whole genome sequencer, Miseq. We identified a list of base pair substitutions (BPSs) and short insertions/deletions (indels) for each samples using the standard software SAMtools.

Results

The average UV-dosage of the selected wells among the 6 lineages was increased over the experimentation (Fig.2). This increase was not only caused by the operation which raised the basic dosage, since the average increased even while the level of basic dosage was steady. This result means that the lineages obtained UV resistance through the experimentation.

The maximum growth rates in the absence of UV exposure of the ancestor and the evolved strains are 0.60±0.1 [h⁻¹] and 0.75±0.1 [h⁻¹] (mean±SD), respectively. Therefore, the growth rates were maintained without drop through the experiment.

Genomic resequencing revealed the number of BPSs and short indels as shown in Table 1. The Ka/Ks, the ratio of non-synonymous substitutions to synonymous substitutions, was 0.53 ± 0.14 lower than 1, suggesting a strong negative selection. That is, the mutations altering protein sequence were basically disfavored through the evolution. In addition, the mutation rate was about 6.0×10^{-8} [bp⁻¹generation⁻¹] which is about 100 times higher than reported value of *E. coli* mutation rate $(5.4\times10^{-10}$ [bp⁻¹generation⁻¹], (Drake, et al. 1998)).

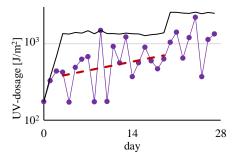


Figure 2: Acceptable UV-dosage increased through the subculture experiment. Daily records of mean UV-dosage on 6 lineages (purple circle plots) oscillated, but the exponential curve fitting (red dashed line) raised even while the basic exposed value (black solid line) was not increased.

BPSs				mutation rate
syn.	non-syn. (stop codon)	indels	Ka/Ks	[bp ⁻¹ generation ⁻¹]
36.8 ± 7.4	64.2 ± 22.5 (5.8 ± 3.0)	3.0 ± 2.7	0.53 ± 0.14	$7.9 \pm 1.6 \times 10^{-8}$

Table 1: Number of detected mutations suggest the existence of selection pressure and high mutation rate. The numbers of BPSs and indels are represented as per genome per lineage [mean±SD, n=6.] Ka/Ks value is adjusted with the ratio of synonymous sites and non-synonymous sites (1:3.24). Mutation rate was calculated considering that the fraction of synonymous mutations in the genome is 21.0 %, and that lineages passed 560 generations through the subculture.

Discussion

UV-dosage of the selected wells were increased even in the period of that the amount of UV-exposure was steady. Therefore it is thought that the lineages obtained UV resistance through the subculture. This consideration suggests that the evolutionary process during the subculture was influenced by UV. Consequently, the subculture environment had about the 100 times higher mutation rates than usual, according to the numbers of BPSs detected on the evolved lineages. In addition, whole genome resequencing revealed the strong negative selection that purges a part of deleterious mutations from the population. The existence of this selective effect is consistent with the fact that the growth rate of the evolved lineages were maintained.

Despite such negative selections purging about 30 non-synonymous mutations estimated from synonymous substitutions, more than 60 non-synonymous BPSs per lineage were detected. Therefore, most mutations were fixed on the *E. coli* genome through the experimentation. Interestingly, the total number of the mutations into stop codon and short indels was 8.8 per lineage (5.8 mutation into stop codon and 3.0 short indels.) It is expected that mutated genes will lose their function, even if part of the sequence keeps its integrity. Therefore, it is suggested that this culture system may be used as an effective tool to accumulate inactivation mutation on the *E. coli* genome, causing genome reduction.

According to the above, we suggest that the fixation and accumulation of many mutations on the *E. coli* genome could be proceeded without growth defects through the growth selection with high mutation rate.

References

Pósfai, G., Plunkett III, G., Fehér, T., Frisch, D., Keil, G. M., Umenhoffer, K., Kolisnychenko, V., Stahl, B., Sharma, S. S., de Arruda, M., Burland, V., Harcum, S. W. and Blattner, F. R. (2006). Emergent Properties of Reduced-Genome *Escherichia coli. Science*, 312:1044-1046

Mira, A., Ochman, H. and Moran, N. A. (2001). Deletional bias and the evolution of bacterial genomes. *Genetics*, 17:589-596

Eyre-Walker, A. and Keightley, P. D. (2007). The distribution of fitness effects of new mutations. *Nature Review Genetics*, 8:610-618

Drake, J. W., Charlesworth, B., Charlesworth, D., and Crow, J. F. (1998). Rates of Spontaneous Mutation. *Genetics*, 148:1667-1686

Task Partitioning based on Response Threshold Model in Robot Harvesting Task

Wonki Lee¹ and DaeEun Kim¹

¹Biological Cybernetics Lab, School of Electrical and Electronic Engineering, Yonsei University, 50 Yonsei-ro, Seodaemun-gu, Seoul, 120-749, South Korea wonkilee@yonsei.ac.kr

In this paper, we invest an effect of task partitioning as the supplement to the navigation using path integration in a harvesting task of a robotic swarm. We present a simple method based on the response threshold model to allocate an individual robot to a partitioned task and show that the task partitioning will increase system performance by reducing the negative effect of path integration errors. The results of proposed method are analyzed and compared to case that task partitioning is not employed to a given task.

We apply an approach of an autonomous task partitioning in a collective harvesting task in a circular arena as shown in Figure 1. The small empty circle at the center of arena is a nest, and the source area where the clustered objects are placed is located in the right side of the nest in a constant distance. All robots stay around a nest at first, and move randomly to search objects. If robots find objects, robots return with them to nest just using path integration as their only means of navigation. Robots put down carrying objects after arriving at nest and they turn direction to move to an object source to harvest other objects also using path integration. Path integration is a process that uses cues to track distance and direction in order to estimate a current position to get home, and is widely used by animals for dead reckoning navigation (Mittelstaedt and Mittelstaedt, 1982).

During the harvesting task, if robots fail to reach the desired destination in the end of navigation, robots move randomly until they arrive at nest or source. If the path integration error is high or the distance of navigation is long, the probability that robots fail to find the nest is high and these wandering behaviors lower an overall system performance. Under these conditions, task partitioning (Jeanne, 1986) will divide harvesting tasks into small tasks with a short distance and reduce the negative effect of path integration errors. There are many examples of task partitioning concerning the collection of food and other materials in insect societies (Ratnieks and Anderson, 1999).

To implement task partitioning, each robot decides its current task based on the response threshold model (Theraulaz et al., 1998). The response threshold model has some similarity with so-called market-based model of task allocation

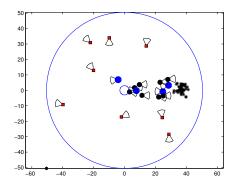


Figure 1: Snapshot of simulation environment. Robots move randomly to harvest objects in a circular arena with a fanshaped detecting range. The dark (red in web site) squares are robots that move randomly to search objects and black (black in web site) stars are clustered objects located on the right side of nest. The large black (blue in web site) circles are robots that return to nest carrying the objects and the small black (black in web site) circles are robots that go to an object source from nest to harvest other objects. All robots use the path integration to move between nest and source.

where agents bid for a given task. In our work, robots switch their current tasks on the way from source to nest after harvesting objects by dropping carrying objects in the current positions if the moving distance D_m of robots is higher than a certain threshold θ_d $(0 < \theta_d < D_m)$ and they detect other robots not carrying objects within a detecting range.

Naturally, robot's threshold of moving distance should be a function of an average moving distance from source to base and the probability that robots detect other robots carrying no objects. Therefore, an optimal threshold for task switching depends on environment components, such as distance between nest and source and density of robots. In addition, the entire environment information is not available to an individual robot, and each robot should choose the best commensurate task with its current states using its own local

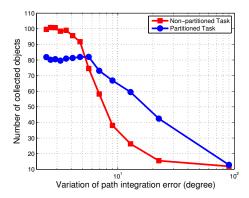


Figure 2: Number of collected objects with variation of path integration error during 5000 simulation time steps.

knowledge alternatively. Under these limitations, choosing an optimal threshold is a complex problem. In our work, we limit ourselves to a simple method for setting this task switching threshold randomly. Threshold value for task partitioning is selected randomly whenever the robot harvests an object and each robot changes its task based on this value.

We compare the system performance by counting the number of collected objects by all robots. Figure 2 shows the averaged results of 20 repetitions with various path integration errors in two strategies, non-partitioned and randomly partitioned. When the path integration direction error is bigger than a certain error (in this paper, $\pm 5.2^{\circ}$ that is related to navigation distance and size of nest), the task partitioned method produces more improved performance. On the contrary to this, if there is small direction errors, navigation returning to base using task partitioning obtains better performance than without task switching. Frequent task switching causes repetitions of dropping and re-harvesting behaviors and these additional behaviors reduce the number of harvesting objects if the path integration error is low.

Figure 3 shows effects of number of robots and navigation distance when the direction error is set to $\pm 9^{\circ}$. As the number of robots is increased, the probability that robots see another robot is increased and task partitioning helps to improve system performances. On the other hand, the number of harvesting objects is decreased as the increasing of navigation distance. In both results, task partitioned navigation shows better results than non-partitioned method.

The goal of our experiments is to investigate whether task partitioning can reduce the negative effects of path integration errors in navigation. We validate our approach using simulation-based experiments and study how task partitioning can improve the performance of our harvesting task. Robots randomly decide the moving distance threshold and the experiments results show that if the path integration error is high, the robots that switch tasks perform better than those that don't, but if the error is low, non-partitioned strat-

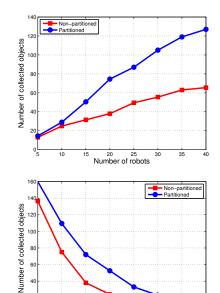


Figure 3: Number of collected objects depend on the number of robots (up) and radius of a circular arena (down).

egy is advantageous. The proposed strategy is very simple and far from being an optimal solution, nevertheless we improved the performance when the path integration error is high. There will be an adaptive way to find more optimal task switching function to achieve better performance, and we will leave it as a future work.

Acknowledgements

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST) (No. 2012R1A2A4A01005677).

References

Jeanne, R. L. (1986). The evolution of the organization of work in social insects. *Monitore Zoologico Italiano-Italian Journal of Zoology*, 20(2):119–133.

Mittelstaedt, H. and Mittelstaedt, M.-L. (1982). Homing by path integration. In *Avian navigation*, pages 290–297. Springer.

Pini, G., Brutschy, A., Birattari, M., and Dorigo, M. (2009). Interference reduction through task partitioning in a robotic swarm. In Sixth International Conference on Informatics in Control, Automation and Robotics—ICINCO, pages 52–59.

Ratnieks, F. L. and Anderson, C. (1999). Task partitioning in insect societies. *Insectes sociaux*, 46(2):95–108.

Theraulaz, G., Bonabeau, E., and Denuebourg, J. (1998). Response threshold reinforcements and division of labour in insect societies. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 265(1393):327–332.

Insect-Inspired Robot Coordination: Foraging and Coverage

Sjriek Alers¹, Karl Tuyls², Bijan Ranjbar-Sahraei¹, Daniel Claes² and Gerhard Weiss¹

¹Maastricht University, P.O. Box 616, 6200 MD, Maastricht, The Netherlands ²University of Liverpool, Ashton Building, Liverpool L69 3BX, United Kingdom

Abstract

In this paper we investigate coordination principles inspired by the behaviour of honeybees and ants for coordination purposes in multi-robot systems. Specifically, we study the problem instances of bee-inspired robot Foraging and ant-inspired robot Coverage, where Foraging is the problem of exploring the environment in search of food or provisions and Coverage is the problem of deploying a robotic swarm in the environment with the task of maximising the sensor coverage of the environment. To effectively and efficiently solve both problems, distributed multi-robot coordination is required. For the first problem we investigate a bee-inspired solution method. The second problem is studied using a stigmergic approach. In an extensive set of experiments we first study the feasibility of the proposed multi-robot coordination for robotic swarms with extended resources and discuss the benefits and limitations of using these swarms. Furthermore, as the downsizing in swarm robotics becomes increasingly important with ongoing miniaturization in various applications, the feasibility of the proposed coordination techniques for robotic swarms with limited resources is studied in detail; the practical requirements for overcoming the limitations of these swarms are introduced and the main need to incorporate these robots in real world experiments is discussed.

Introduction

Swarm robotic systems are motivated by a wide range of application areas, such as for instance surveillance and patrolling, where mobile guarding robots are considered as an alternative and improvement over fixed security cameras and even humans. Other application areas include exploration and identification of hazardous environments (e.g., nuclear plants and fire detection), mobile sensor networks, space exploration, etc.

Recent years have seen an increasing interest in taking inspirations from natural phenomena for solving computational problems in disciplines at the intersection of computer science, robotics and economics. An interesting natural phenomenon is the behaviour that can be observed in colonies of social insects such as ants and bees. For instance, recent work shows a strong potential in creating artificial systems that mimic insect behaviour that can solve complex coordination tasks such as e.g., routing on the internet, mobile

ad hoc network routing, robotic tasks, etc. (Lemmens and Tuyls, 2012; Dressler and Akan, 2010; Floreano and Mattiussi, 2008). These insects have evolved over a long period of time and display remarkable behaviours that are highly suitable for addressing the complex tasks that they are facing. Swarm optimisation algorithms, like ant colony optimisation (Dorigo et al., 2006), rely on pheromone trails to mediate (indirect) communication between agents.

These pheromones need to be deposited and sensed by agents while they decay over time. Though easy to simulate, artificial pheromones are hard to bring into real-life robotic applications. However, recently non-pheromone-based algorithms where developed as well (Lemmens, 2011). Such algorithms are inspired by the foraging and nest-site selection behaviour of (mainly) bees. In general, bees explore the environment in search for high quality food sources and once returned to the hive they start to dance in order to communicate the location of the source. Using this dance, bees recruit other colony members for a specific food source.

We draw inspiration from these insect behaviours with the goal to create emergent intelligent systems for distributed coordination that can be deployed in real world settings. However, doing the experiments with real robotic swarms is very challenging. On one side robotic swarms with limited resources such as e-pucks (Mondada et al., 2009) are robust, compact, easy to setup and relatively cheap in terms of price but have limited sensing, computation and actuation capabilities. On the other hand robotic swarms with extended resources, which contain advanced cameras and general purpose computers, have high computational power, good movement capabilities and are modular. In this paper, we provide an extensive study of the pros and cons of each of these swarm types.

The remainder of the paper is organised as follows. First we introduce the biological background of the foraging and coverage approaches. Then we continue with the swarm robotics approach with extended resources and it is explained how multi-robot coordination can be implemented in these swarms in a set of experiments, highlighting the vision and communication capabilities of such advanced robots.

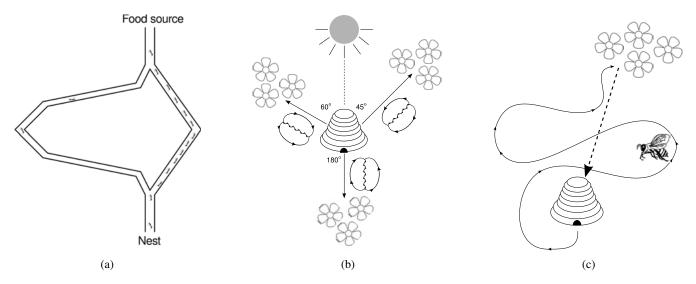


Figure 1: Biological inspiration (a) ants exploring two paths; the shortest path prevails. (b) honeybee waggle dance communicating a PI vector. (c) Lévy flight and path integration.

After that we continue by elaborating the coordination approaches for swarm robotics with limited resources: vision and communication techniques are explained; implementation of both multi-robot coverage and foraging are presented in a set of experiments in the context of limited available resources, highlighting the novel techniques in computation and communication that are used to make such implementations possible. Finally we conclude.

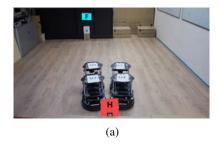
Biological Inspiration

Most of the research in swarm intelligence revolves around the behaviour of ants (Dorigo and Stützle, 2004; Dorigo and Blumb, 2005; Dorigo et al., 2006). The principle is simple yet elegant: ants deposit a pheromone trail on the path they take during travel. Using this trail, they are able to navigate toward their nest or food and communicate with their peers. More specifically, ants employ an indirect recruitment strategy by accumulating pheromone trails. When a trail gets strong enough, other ants are attracted to it and will follow this trail toward a food destination. The more ants follow a trail, the more pheromone is accumulated and in turn the trail becomes more attractive for being followed. This is known as the autocatalytic process. Since long paths take more time to traverse, it will require more ants to sustain a long path. As a consequence, short paths will eventually prevail, see Figure 1(a). Pheromone-based algorithms are already used to address various problems successfully, such as (amongst others) the Traveling Salesman Problem (Dorigo and Stützle, 2004), Routing Problem (Di Caro et al., 2005), Group Shop Scheduling (Blum and Sampels, 2004), and area coverage with robots (Wagner et al., 1999; Ranjbar-Sahraei et al., 2012b).

Foraging honeybees display two types of behaviour, i.e.,

recruitment and navigation. In order to recruit other colony members for food sources, honeybees inform their nest mates of the distance and direction of these food sources by means of a waggling dance performed on the vertical combs in the hive. This dance (i.e., the bee language) consists of a series of alternating left-hand and right-hand loops, interspersed by a segment in which the bee waggles her abdomen from side to side. The duration of the waggle phase is a measure of the distance to the food. The angle between the sun and the axis of a bees waggle segment on the vertical comb, represents the azimuthal angle between the sun and a target location, i.e., the direction in which a recruit should fly (see Figure 1(b) and 1(c)). Other members of the colony can adopt the "advertisement" for a food source. The decision mechanism for adopting an "advertised" food-source location by a potential recruit is not completely understood. It is considered that the recruitment amongst bees is a function of the quality of the food source.

Different species of social insects, such as honeybees and desert ants, make use of non-pheromone-based navigation. Non-pheromone-based navigation mainly consists of Path Integration (PI), which is the continuous update of a vector by integrating all angles steered, and all distances covered (Collett et al., 1998). A PI vector represents the insects knowledge of direction and distance towards its destination. To construct a PI vector, the insect does not use a mathematical vector summation, but employs a computationally simple approximation (Collett et al., 1998). Using this approximation, the insect is able to return to its destination directly. More precisely, when the path is unobstructed, the insect solves the problem optimally. However, when the path is obstructed, the insect has to fall back on other strategies such as exploration or landmark navigation (Cheng et al., 1987;



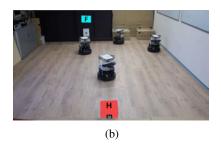




Figure 2: Multi-Robot foraging using swarm robots with extended resources. (a) All robots start at the hive (H) location. (b) Robots are exploring the unknown environment randomly. The left two robots have found the food (F) location and are foraging between the hive and the food location. (c) All robots have converged to foraging behavior.

Collett et al., 2002) to solve the problem. Obviously, bees are able to fly, i.e., when they encounter an obstacle, they can mostly choose to fly over it. However, even if the path is unobstructed, bees tend to navigate over the entire path using landmarks. The landmarks divide the entire path into segments and each landmark has a PI vector associated with it. This behaviour decreases navigation errors and ensures robustness. We refer to a home-pointing PI vector as a Home Vector (HV). PI is used in both exploration and exploitation. During exploration insects constantly update their HV. It is however, not used as an exploration strategy. During exploitation, the insects update both their HV and the PI vector indicating the food source, and use these vectors as guidance to a destination.

Swarm Robotics with Extended Resources

In this section we introduce the swarm robotics with extended resources. These swarms use general purpose computers, high quality and advanced video cameras, 3D sensors for mapping (e.g., laser range finders), accurate wheel encoders that makes enhanced odometry possible, fused data of accelerometers, and a gyroscope.

The Turtlebot¹ platform is a robot with extended resources. This robot is equipped with a laptop with core-i3 CPU for computation that is running the Robot Operating System² framework.

As a main sensing unit the Turtlebot is equipped with a Kinect sensor. The full RGBD information is used to detect and locate AR markers. For static obstacle detection, we only use the depth information of the sensor together with three bumpers that are located in the front half of the robot. Furthermore, the robot estimates its position by integrating the wheel odometry and gyro information. Hence, no map of the environment is built and the only known reference point is the target location marker. This can lead to the problem that if the odometry is faulty, the robot does not always find the target location back. As a solution the robots fall back into a search mode, if this is the case. Another solution could

be to implement a Northstar like navigation system, by providing a fixed frame of reference which is almost always visible from any location.

Robot Vision and Communication

To enable visual robot-robot detection we equipped every Turtlebot with six unique markers, which are oriented in a way that at least one marker is visible from any angle. To track and decode these markers we make use of a toolkit called ALVAR, more specifically we use the ROS wrapper³ of this library. We use a customised bundle detection method to determine the center of the detected robot dependent on the decoded markers. Kalman filtering is applied to get better and more stable readings and consequently a more accurate estimate of the detected robots position, heading and speed. These parameters are used again for collision avoidance.

Communication is realised over Wi-Fi with a UDP connection to each Turtlebot using the LCM library⁴. Even though global communication would be possible, we limit the communication, such that every robot listens only to its own channel. To simulate local communication, the robots can only communicate with another robot when it is in view and in close proximity, i.e., less than one meter away.

Experiments

In this subsection we briefly describe the practical algorithms needed for implementation of bee foraging on swarms with extended resources. This clearly highlights the main benefit of using these type of robots which is the possibility of implementation of very advanced algorithms in a very convenient way.

Collision Avoidance In order to avoid robot to robot collisions, we rely on the marker detection to predict positions and speeds of the other robots. This information can be used to efficiently compute a non-colliding speed vector as we

http://www.Turtlebot.com/

²http://www.ros.org/

³http://wiki.ros.org/ar_track_alvar

⁴https://code.google.com/p/lcm/

have developed previously in (Claes et al., 2012). In contrast to this previous approach, in which the robot-robot detection was avoided by using a global reference frame and broadcasting the positions to all robots via Wi-Fi, solely the marker detection and the predictions using a Kalman filter are used. This means that a few collisions still might occur due to failure to detect the markers of the other robots and additionally, there are certain configurations in which the robots cannot see each other due to the field of view of the Kinect sensor, e.g., when two robots drive in a V-shape towards each other, the field of view of the Kinect is too narrow to detect the other robot.

As shown in the previous work of authors in (Alers et al., 2014), multiple Turtlebots perform a foraging task, i.e., starting at the Hive (H) location and randomly exploring the unknown environment for a specific Food (F) location. This is shown in Figure 2. Another way of locating a food location is by asking bypassing robots for a known food location, which is done by simulating local communication over Wi-Fi. When the source is found the robot starts to exploit this source, i.e. driving from the food to the hive location until the food is depleted or a better source is found. A video showing this demonstration can be found in the online material⁵.

Discussion

In summary, the robotic swarms with extended resources can accomplish many tasks successfully. These swarms use more sophisticated sensors like RGBD and VGA camera's to detect environmental features and can communicate with each other using Wi-Fi, while they use vision processing to simulate local communication.

Robots with extended resources have their up and downside. The downside of these robots are that they are bigger than the robots with limited resources; they are more expensive, as the sensors and computational units are more costly. These type of robots run on an operating system that has a steep learning curve. On the other hand, the upside of such robots is that they are more versatile. It is much easier to extend the platform with new sensors by for example plugging in an additional camera, or a dedicated control unit into the usb port. The computational limitations are not restrictive, e.g., image processing can be easily done without exhausting any other resource. Last but not least, the software modules can be easily reused or shared with the robotic community, as all the modules are developed in a standardized way.

Next we study the swarms of robots with limited resources which are simple, compact, small and relatively cheap. These robots are very robust by using lots of proven technologies (e.g., microcontrollers, basic sensors and actuators)

Table 1: E-puck technical specification

Element	Technical information	
Processor	dsPIC30F6014A @ 60 MHz (15 MIPS),	
	16-bit microcontroller with DSP core	
Memory	RAM: 8KB Flash: 144 KB	
Motors	2 stepper motors with a 50:1 reduction gear	
Camera	VGA color camera with resolution of 640×480 pixels	
	o to // too pinets	
LEDs	8 red LEDs on the ring, green LEDs on the body, 1 high intensity red LED in the front	
Wireless	Bluetooth for robot-computer and	
Communication	,	
	Infrared for robot-robot communication	

Swarm Robotics with Limited Resources

In this section we introduce the swarm robotics with limited resources approach, which refers to the swarms of relatively small and cheap robots that have limited computation power (i.e., embedded micro-controllers), limited memory and very simple sensors and basic communications. Such swarms are in contrast to the swarms with extended resources that take advantage of powerful computers, advanced cameras and Wi-Fi communication capabilities.

The e-puck platform is an example of a robot with limited resources. e-puck is a small robot for educational and research purposes, developed by the EPFL University (Mondada et al., 2009). This robot is efficiently used in numerous projects in the domain of swarm robotics and swarm intelligence (e.g., works by Alers et al. (2011, 2013a,b); Lemmens et al. (2011); Breitenmoser et al. (2010); Mondada et al. (2009); Ranjbar-Sahraei et al. (2013b)). The main features of the e-puck robot include, but are not limited to; a robust design, flexibility for a large spectrum of educational activities, compact size, and rich on-board accessibilities (e.g., microphones, accelerometer and camera).

The e-puck hardware consists of different sensor types for detecting visible or Infra Red (IR) light, sound, acceleration, etc. The motors are the only actuators which are available in e-puck. A microprocessor of PIC family with 8 KB RAM memory assists the robot to get data from its sensors, analyse it, and perform actions. The main hardware elements of the e-puck robot are listed in Table 1.

As listed in the table, the on-board camera of the e-puck has a resolution of 640×480 pixels, although due to the limited resources, the robot is only capable of storing and processing an image of 40×40 pixels. It is placed at the front of the e-puck, 2.7 cm above the floor. With this camera, objects that are placed on the floor can be detected at a minimum distance of 7.4 cm. The camera angle is approximately 40° , and at this minimum distance, objects of 5.1 cm

⁵http://swarmlab.unimaas.nl/papers/ aamas-2014-foraging

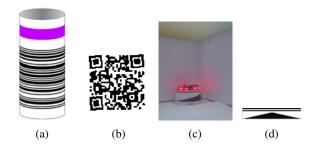


Figure 3: Detectable features presented in (Alers et al., 2013b) (a) Landmarks with barcode. (b) QR-code level 3. (c) Robot LEDs. (d) Robot orientation pattern.



Figure 4: Scenario for validation of robotic vision.

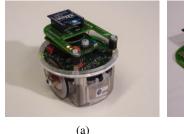
width can be fully monitored.

Robot Vision

Alers et al. (2013b) explored several visual features that can be used for acquiring information from the environment by a robot with limited computational abilities. In this work, for detecting key locations in the environment, such as corners in a maze, the usage of specific landmarks for these locations is investigated. Each landmark consists of an upper ring with a solid color, so that it can be detected from a distance, and on the lower part a unique barcode for keeping track of the landmark numbers, as can be seen in Figure 3(a).

Furthermore, the possibility to detect markers with an even higher data density, QR-codes as in Figure 3(b), are explored. The challenge in the detection of these two-dimensional codes, lies in analysing and processing the camera data with the limited processing and memory resources that are available in the robotic platform.

Finally, the most common feature already available in every swarm robotic setting is explored: the presence of an other robot. It's always favourable to detect the relative distance and orientation to other robots in respect of one's own position. Therefore, the available LEDs on the robot provide a very good feature for robot detection from a distance, see Figure 3(c). Moreover, a specific gradient pattern for nearby robot detection, as shown in Figure 3(d), is designed. This pattern results in a very accurate orientation and distance detection.



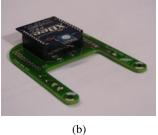


Figure 5: e-puck robot with improved local communication (a) An e-puck robot as used in the experiments. (b) The XBee communication extension board.

A video of this performed experiment on a validation scenario as shown in Figure 4 can be found online⁶, including the intermediate image data from the robot.

Communication

Direct Communication The e-puck robots contain 4 types of sensors with which they have to make sense of the world around them. Namely, 8 IR sensors, 1 camera, 3 microphones, and an accelerometer. Of these sensors, we only use the 8 IR sensors for proximity detection. Additionally, they have two main communication possibilities, namely, IR communication and (limited) Bluetooth communication. The former is prone to interference, its operation is CPU intensive, and as such is only viable for local short-message communication. The latter is limited to 3 simultaneous connections and setting up a single connection can take as long as 10 s. Moreover, 3 simultaneous connections can only be set up in the form of 1 master and 3 slaves. This severely limits communication possibilities.

In order to overcome the shortcomings of the current sensors, we were inspired by the communication module on the AdMoVeo robot designed by Alers and Hu (2009) and have designed an XBee extension board for the e-puck to improve local communication. Its design features are robustness, speed, low power usage, and ease of use. Figures 5(a) and 5(b) shows the e-puck equipped with the extension board and the extension board alone, respectively. XBee ensures reliable, fast, local, peer to peer communication. Moreover, it also provides the possibilities for creating a mesh network between multiple XBee chips. This opens up research possibilities in fields such as Mobile Adhoc Routing.

Stigmergic Communication Inspired by the stigmergic type of communication in ant colonies, robots can get benefits of stigmergy in communication-limited environments. However, despite of a few reports of using chemicals or radio frequency identification tags in robotic experiments by

⁶http://swarmlab.unimaas.nl/papers/adaptive-2013-demo



Figure 6: Darkroom with glow-in-the-dark floor, where the e-puck robots circle around and emit UV light onto the floor.

Fujisawa et al. (2008); Herianto et al. (2007); Johansson and Saffiotti (2009), due to difficulties in implementation and limited extendibility, this approach did not provide sufficient applicability in swarm scenarios. Therefore, motivated by the technique proposed by Kronemann and Hafner (2010), we have designed a test-bed which provides stigmergic communication to the robots as shown in Figure 6.

In this setting the floor is covered by a glow-in-the-dark foil (i.e. a foil covered by phosphorescent material which absorbs UV light and re-emits the absorbed light at a lower intensity for up to several minutes after the original excitation), and robots are equipped with UV-LEDs pointing toward the floor. Therefore, as robots move around they leave glowing trails behind themselves. Furthermore, for detection of these trails, in contrast to the simple method used by Kronemann and Hafner (2010), in which photo-sensors were used to detect glowing trails, we take advantage of the e-puck on-board camera. By capturing an image and applying a green filter to it, we extract the exact pattern of green trails in the image. The patterns in the image can be used to measure the presence of trail and also its density over different locations. Finally, the IR sensors are used for obstacle avoidance⁷.

Experiments

In this subsection we describe the experiments on robotic swarms with limited resources, for two different scenarios of bee foraging and environment coverage, highlighting the benefits of using these type of swarms and practical approaches to overcome their limitations.

Bee Foraging The bee foraging experiments show the effectiveness of the embodied foraging behaviour in a swarm of e-pucks. In Figure 7, we present the stages that the experiment goes through. The goal of the experiment is to show that each separate behaviour actually works in an embodied swarm. Therefore, the experiment starts with a swarm of

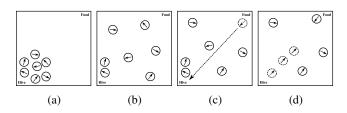


Figure 7: The four stages of Biomimicry Foraging (a) All robots start at the nest location. (b) The robots randomly disperse trough the environment looking for a food location. (c) A robot that has found food returns to the nest location by the shortest possible path. (d) The food location is communicated to other robots and they start to exploit this food-source.

e-pucks surrounding the hive, see Figure 7(a). Figure 7(b) shows the stage in which a portion of the swarm starts foraging while others remain around the hive, waiting for information to exploit. Figure 7(c) presents the situation in which an exploring e-puck finds food and returns to the hive by using its constructed PI vector. Once returned to the nest, the e-puck communicates its PI findings by means of a virtual dance. The hive collects these experiences and offers these to recruits. Finally, Figure 7(d) gives the situation in which other e-pucks communicated with the hive and have attained the PI vector towards the food source and are traveling to the food source guided by this PI vector. A demonstration movie can be found online⁸.

Environment Coverage The multi-robot coverage experiment can be used for various monitoring, rescue, and patrolling missions. Ranjbar-Sahraei et al. (2012b) proposed an stigmergic coverage approach called StiCo which does not need a priori knowledge of the environment, communication among robots or distance measurements. StiCo works based on a very simple motion policy: Each robot circles around with a fixed radius and marks its path with evaporable markers, which denote the borders of robot's territory. Simultaneously, if a robot detects a trail while circling around, it changes its circling direction immediately. This behavior is illustrated in Figures 8(a)-8(c). Ranjbar-Sahraei et al. (2012a) used computer simulations to show that StiCo is very simple, but efficient, robust and even extendable. An illustration of the StiCo coverage approach with real robots is shown in Figures 8(d)-8(f) (Ranjbar-Sahraei et al., 2013b).

Discussion

In summary, the robotic swarms with limited resources can accomplish many tasks successfully. These swarms can use their simple cameras to detect environmental features (shown in Figure 3) and they can communicate using IR

⁷More technical details available in http://swarmlab.unimaas.nl/stico/indoor-experiments.

⁸http://swarmlab.unimaas.nl/papers/ bnaic2011demo/

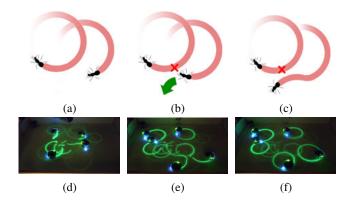


Figure 8: StiCo coordination principle: (a) robots circle around. (b) the right robot detects pheromone. (c) the right robot changes circling direction. (d)-(f) Vision-based Stigmergic Coverage using glowing trails.

communication or XBees (shown in Figure 5). Their small size makes it possible to do experiments in very compact environments (e.g. the testbed shown in Figure 6).

Robotic swarms with limited resources have up and downsides. The upside of these kind of robots is that they are very robust by using lots of proven technologies (e.g., microcontrollers and basic sensors and actuators). They are also relatively cheap to make which for a large swarm could be a more realistic scenario. The downside of using limited resources is that the possibilities of the robotic system is constraint by the hardware platform and the computational possibilities. Also the use of relatively basic sensors limits these kind of platforms in perceiving the environment and interacting or communicating with it. It is also hard to extend the platform with new sensors, which would require some electrical engineering.

We experienced that there are some difficulties, in doing intensive experiments with these swarms: These robots don't contain a modular structure such that in terms of damage, one can replace the damaged part with a new one. Besides, the programmers should usually code all the requirements (e.g, image processing modules) which makes it very time consuming to implement all the requirements and hard to debug.

Concluding Remarks

In this paper we investigated two fundamental problems in swarm robotics, the Foraging and Coverage, from a multirobot coordination perspective. For the former problem a bee-inspired solution was introduced while pheromone-based communication was used to address the latter problem. For implementation of such problems on real world robotic swarms first a robotic platform with extended resources, Turtlebot, was introduced and the practical requirements to implement the foraging algorithm on a swarm of these robots were discussed. Afterwards, a robotic plat-

form with limited resources, e-puck, was introduced. It was shown how limitations of these kind of robots such as limitation in computational power and low quality vision can be overcome; possible extensions for direct robot-robot communication and the indirect stigmergic communication were considered.

Although the main disadvantage for robots with extended resources is that they are still more expensive and bigger in size and the main disadvantage of the robots with limited resources is that their possibilities are too limited. One can argue that in the near future these argumentations are not relevant anymore. From the current development of System on a Chip (SoC) controllers, used for mobile phones and tablets, one can already see that these low power processors increase processing strength every year, are not really costly and are widely available. Also several sensors and RGBD cameras are miniaturised and will eventually turn up inside a tablet or smartphone. These sensors and controllers will suit the needs of real swarm robotic applications and validate the current use of robots with both limited and extended resources within the field of swarm robotic research.

Considering the pros and cons of each type of studied robotic swarm, one can think of heterogeneous swarms that employ both type of robots in the same mission. This is already explored by the authors in (Ranjbar-Sahraei et al., 2013a) and is still one of the main directions for their future works.

References

- Alers, S., Bloembergen, D., Hennes, D., de Jong, S., Kaisers, M., Lemmens, N., Tuyls, K., and Weiss, G. (2011). Bee-inspired foraging in an embodied swarm. In *Proceedings of the Tenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1311–1312.
- Alers, S., Claes, D., Tuyls, K., and Weiss, G. (2014). Biologically inspired multi-robot foraging. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems (AAMAS)*, page (to appear). International Foundation for Autonomous Agents and Multiagent Systems.
- Alers, S. and Hu, J. (2009). Admoveo: A robotic platform for teaching creative programming to designers. In Proceedings of the 4th International Conference on E-Learning and Games: Learning by Playing. Game-based Education System Design and Development (Edutainment), Edutainment '09, pages 410–421, Berlin, Heidelberg. Springer-Verlag.
- Alers, S., Ranjbar-Sahraei, B., May, S., Tuyls, K., and Weiss, G. (2013a). Evaluation of an experimental framework for exploiting vision in swarm robotics. In *Advances in Artificial Life, ECAL*, volume 12, pages 775–782.
- Alers, S., Ranjbar-Sahraei, B., May, S., Tuyls, K., and Weiss, G. (2013b). An experimental framework for exploiting vision in swarm robotics. In ADAPTIVE 2013, The Fifth International Conference on Adaptive and Self-Adaptive Systems and Applications, pages 83–88.

- Blum, C. and Sampels, M. (2004). An ant colony optimization algorithm for shop scheduling problems. *Journal of Mathematical Modelling and Algorithms*, 3(3):285–308.
- Breitenmoser, A., Schwager, M., Metzger, J., Siegwart, R., and Rus, D. (2010). Voronoi coverage of non-convex environments with a group of networked robots. In *IEEE Interna*tional Conference on Robotics and Automation (ICRA), pages 4982–4989.
- Cheng, K., Collett, T., Pickhard, A., and Wehner, R. (1987). The use of visual landmarks by honeybees: Bees weight landmarks according to their distance from the goal. *Journal of Comparative Physiology A*, 161(3):469–475.
- Claes, D., Hennes, D., Tuyls, K., and Meeussen, W. (2012). Collision avoidance under bounded localization uncertainty. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012), Vilamoura, Portugal.
- Collett, M., Collett, T. S., Bisch, S., and Wehner, R. (1998). Local and global vectors in desert ant navigation. *Nature*, 394(6690):269–272.
- Collett, M., Harland, D., and Collett, T. S. (2002). The use of landmarks and panoramic context in the performance of local vectors by navigating honeybees. *The Journal of Experimental Biology*, 205:807–814.
- Di Caro, G., Ducatelle, F., and Gambardella, L. M. (2005). Swarm intelligence for routing in mobile ad hoc networks. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 76–83.
- Dorigo, M., Birattari, M., and Stutzle, T. (2006). Ant colony optimization: Artificial ants as a computational intelligence technique. Computational Intelligence Magazine, IEEE, 1(4):28–39
- Dorigo, M. and Blumb, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344:243–278.
- Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. A Bradford book. BRADFORD BOOK.
- Dressler, F. and Akan, O. B. (2010). A survey on bio-inspired networking. *Computer Networks*, 54(6):881–900.
- Floreano, D. and Mattiussi, C. (2008). Bio-inspired artificial intelligence: theories, methods, and technologies. The MIT Press
- Fujisawa, R., Imamura, H., Hashimoto, T., and Matsuno, F. (2008). Communication using pheromone field for multiple robots. In Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on, pages 1391 –1396.

- Herianto, Sakakibara, T., and Kurabayashi, D. (2007). Artificial pheromone system using rfid for navigation of autonomous robots. *Journal of Bionic Engineering*, 4(4):245 253.
- Johansson, R. and Saffiotti, A. (2009). Navigating by stigmergy: A realization on an rfid floor for minimalistic robots. In Robotics and Automation, 2009. ICRA '09. IEEE International Conference on, pages 245–252.
- Kronemann, M. L. and Hafner, V. V. (2010). Lumibots making emergence graspable in a swarm of robots. In *The ACM Designing Interactive Systems Conference*, pages 408–411.
- Lemmens, N. (2011). Bee-inspired Distributed Optimization. Maastricht University.
- Lemmens, N., Alers, S., and Tuyls, K. (2011). Bee-inspired foraging in a real-life autonomous robot collective. In *Proceedings of the 23rd Benelux Conference on Artificial Intelligence* (BNAIC), pages 459–460.
- Lemmens, N. and Tuyls, K. (2012). Stigmergic landmark optimization. *Advances in Complex Systems*, 15(8).
- Mondada, F., Bonani, M., et al. (2009). The e-puck, a robot designed for education in engineering. In *9th Conference on Autonomous Robot Systems and Competitions*, volume 1, pages 59–65. IPCB: Instituto Politcnico de Castelo Branco.
- Ranjbar-Sahraei, B., Alers, S., Stankova, K., Tuyl, K., and Weiss., G. (2013a). Towards soft heterogeneity in robotic swarms. In *Proceedings of the 25th Benelux Conference on Artificial Intelligence (BNAIC)*, pages 384–385.
- Ranjbar-Sahraei, B., Alers, S., Tuyls, K., and Weiss, G. (2013b). Stico in action. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1403–1404. International Foundation for Autonomous Agents and Multiagent Systems.
- Ranjbar-Sahraei, B., Weiss, G., and Nakisaee, A. (2012a). A multi-robot coverage approach based on stigmergic communication. In *Multiagent System Technologies*, volume 7598 of *Lecture Notes in Computer Science*, pages 126–138. Springer.
- Ranjbar-Sahraei, B., Weiss, G., and Nakisaee, A. (2012b). Stigmergic coverage algorithm for multi-robot systems (demonstration). In *Proceedings of the Eleventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 3, pages 1497–1498.
- Wagner, I. A., Lindenbaum, M., and Bruckstein, A. M. (1999). Distributed covering by ant-robots using evaporating traces. *Robotics and Automation, IEEE Transactions on*, 15(5):918–933.

Designing a Robotic Platform Controlled by Cultured Neural Cells

Norihiro Maruyama^{1*}, Atsushi Masumori^{1*}, Julien Hubert¹, Takeshi Mita¹, Douglas Bakkum², Hirokazu Takahashi¹ and Takashi Ikegami¹

¹The University of Tokyo

²ETH Zurich

*These authors contributed equally to this work.
maruyama@sacral.c.u-tokyo.ac.jp

Abstract

Robot experiments using real cultured neural cells as controllers are a way to explore the idea of embodied cognition. Real cultured neural cells have innate plasticity and a sensory motor coupling is expected to develop the neural circuit. We designed a system in which a robot moving in a real environment is controlled by cultured neural cells growing on a glass plate attached to a High-Density Microelectrode CMOS Array(HDMEA). The IR sensors on a robot will feedback onto the neural cells through HDMEA and the activity of the neural cells will be read again by HDMEA and sent back to determine the speed of the robot. Most of the previous works have used the relatively low-density multi-electrode array for recording and stimulating the neural assembly. Our system has the advantage of a high-density spatial and temporal array so that we can precisely detect which neurons get fired and suppressed. A preliminary finding from the experiment is that synchronized neural activation is retained in cultured neurons even after detached from a robot.

Introduction

Recently, it became easier and popular to study the coupling between a robot and a network of cultured neural cells. In those studies, the sensory information coming from the moving robot is used to stimulate the neural cells, and the resulting activities of those determine the speed of the motors driving the robot. This is what is called a "closed loop" experiment. We believe that it is critical to conduct such closed loop experiment for revealing biological memory and adaptability with respect to embodiment. Behavior is not a one way function of sensory inputs but behavior assimilates by itself.

For example, Bakkum et al. (2008) have proposed a new method to train a biological neural cells to achieve a desired pattern for multiple stimulus. Kudoh et al. (2008) have proposed another learning method using a cultured neural system that incrementally learns to respond in a particular way to a particular input. One drawbacks of those studies is that their microelectrode array has not enough space resolution, so that it is difficult to stimulate/detect a single neuronal state. The other drawback is giving an external evaluation function that enables a coupled neuro-robot

system to work. Such evaluation function should be developed from the neuro-robot itself, namely, we have to develop neural self-organization of sense-making behavior with a mobile body. In order to overcome those drawbacks, we use a recently developed high density CMOS array (HD-MEA) capable of detecting the activity of individual neurons with high precision. With HMDEA, we can measure the spatio-temporal neural pattern with a higher precision and reveal how neural plasticity and memory can self-organize the sense-making behavior in a given environment.

Method

A simplest task we seek here is avoiding or reaching behavior of the robot without putting further constraints.

The main components of this system are the HDMEA monitoring the culture of neural cells, the robot in its arena and the interface connecting them. The system gathers the signal from the robot, stimulates the neural cells by using HMDEA and sends the motor output signal to the robot. This way, the robot and the neural cells form a closed loop. The HDMEA we used in this study provides a higher spatio-temporal resolution compared to previous studies [Frey et al. (2010)]. Thus we can monitor all the neural activities by using less than 126 cultured neural cells and the adequate electrode channels. For the moment, we can stimulate at most two cells at a time out of 126 channels available. The sampling rate of HDMEA is 20kHz. A software MeaBench developed by D. Wagenaar [Wagenaar et al. (2005)] is used for recording and detecting spikes and controlling the whole system.

We used Elisa-3 (Manufactured by CGtronic) as a mobile robot. Elisa-3 is a circular small robot of 2.5 cm radius and it has two independently controllable wheels. In this experiment, we use the front right and front left distance sensors as sensory stimulation for the neural cells.

We chose two excitatory neurons as left or right inputneuron for receiving the stimuli. Stimulation to the neural cells is determined by the in-take sensor inputs of this robot. We designed it as that the closer a robot approaches a wall and the higher the sensory inputs become, the more frequently the neurons are stimulated.

We selected 20 neurons as output-neuron within the vicinity of each input-neuron for computing each left and right motor outputs. The left and right wheel speeds are computed based on the number of firing of the output-neurons that is integrated every 100ms. More practically, we compute the left and right wheel speeds V_l and V_r as follows:

$$V_{l,r} = \sum_{i \in \mathbf{N}_{1,\mathbf{r}}} \omega_i v_i + C$$

These virtual neural states v_i take the positive integers which are equal to the number of spiked neurons over a given time interval, and summed up with the fixed weight ω_i . Finally a positive constant C is added. N_1 and N_r are the number of left and right output-neurons.

In this time, as ω_i is negative value and C is positive integer, the robot move forward when output-neurons are not active and the higher the activities of output-neurons becomes, the slower the speed of the forwarding and finally robot move back. The cultured neurons were prepared from the cerebral cortex of fetus of Wister rats.

Result

We conducted 10 minutes robotic experiments in a 60cm by 60cm arena and recorded both the neural activity and the behavior of the robot. The neural spiking patterns are recorded in the pre-experiment, during the experiment, post-experiment and an hour later after the experiment. They are shown in Figure 1. The neural activity of the pre- and post-experiment are different and it seems that the influence of the closed-loop experiment lasts at least for one hour after the experiment.

We then evaluated the temporal changes of the wall avoidance frequency (how many times a robot can avoid crashing into a wall) and computed the time evolution of the correlation coefficient between the neural firing rates in Figure 2. It is indicated that the correlation is inversely correlated with the wall avoidance frequency.

Discussion

We report the design of a robot experiment with a cultured neuron using HDMEA and the preliminary result of the experiment. The results indicate that the correlation of activities of each neurons is inversely correlated with the wall avoidance frequency. Furthermore, we observed that the effect of the stimulation remains for at least 1h after the experiment. It is interesting in terms of memory or learning in the neural network. However, as these are the preliminary results, we need more experiments and analyses.

The system we proposed has a high-resolution that can monitor individual neuronal activities. Therefore, we expect the more detailed investigation about relationship of dynamics of neural network, memory or learning with embodiment. We plan to do more detailed studies such as analyzing

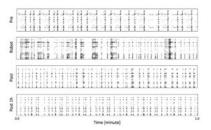


Figure 1: Raster plot of the neural activities of preexperiment(top), during the robot experiment (2nd), the post experiment(3rd) and the post one hour (bottom). The vertical axis is the index of neurons and the horizontal axis is time steps (of 1 minutes tics).

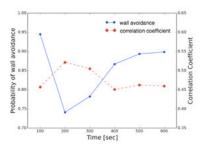


Figure 2: The probability of wall avoidance of robot (blue) and the correlation coefficient of the neural activity (red).

temporal-spatial patterns of neural activities or estimating functional aspects of the network.

Acknowledgements

The CMOS Array used in this study is provided by Prof. Andreas Hierlemann, ETH Zurich.

References

Bakkum, D. J., Chao, Z. C., and Potter, S. M. (2008). Spatiotemporal electrical stimuli shape behavior of an embodied cortical network in a goal-directed learning task. *Journal of neural engineering*, 5(3):310.

Frey, U., Sedivy, J., Heer, F., Pedron, R., Ballini, M., Mueller, J., Bakkum, D., Hafizovic, S., Faraci, F. D., Greve, F., et al. (2010). Switch-matrix-based high-density microelectrode array in cmos technology. *Solid-State Circuits, IEEE Journal of*, 45(2):467–482.

Kudoh, S. N., Tokuda, M., Kiyohara, A., Hosokawa, C., Taguchi, T., and Hayashi, I. (2008). Vitroid-a robot with link between living neuronal network in vitro and robot body. In *Mechatronics and Automation*, 2008. ICMA 2008. IEEE International Conference on, pages 375–378. IEEE.

Wagenaar, D., DeMarse, T. B., and Potter, S. M. (2005). Meabench: A toolset for multi-electrode data acquisition and on-line analysis. In *Neural Engineering*, 2005. Conference Proceedings. 2nd International IEEE EMBS Conference on, pages 518–521. IEEE.

Neuroevolution of sequential behavior in multi-goal navigation task

Sergey Muratov¹, Konstantin Lakhman² and Mikhail Burtsev²

¹Moscow Institute of Physics and Technology (State University), 9 Institutskiy per., Dolgoprudny, Russia
²National Research Center "Kurchatov Institute", 1 Kurchatov sqr., Moscow, Russia
klakhman@gmail.com

Abstract

In this paper we consider a problem of evolutionary generation of behavioral sequences for a mobile robot in the environment with multiple goals. Many real-world tasks can be abstracted in terms of generation of actions sequences. To address this problem we utilize a neuroevolutionary algorithm, based on a neuron duplication, as a method to generate the robot's controller for multi-goal sequential navigation. We show that neuroevolution produces stable behavioral strategies that can be successfully utilized even in the case of changing or randomized environment.

Introduction

Generation of goal-directed behavioral sequences is one of the challenging problems in the field of robotics. In spite of significant success in solving this problem for the single goal scenario there is still a little progress in extending working solutions to the multi-goal tasks.

Navigation is one of the basic tasks in the field of autonomous robot control. A wide range of different approaches is employed to solve this problem such as reinforcement learning Smart and Kaelbling (2002), Bayesian methods (Cassandra et al., 1996; Martinez-Cantin et al., 2009), artificial neural networks (Janglova, 2004) and many others including hybrid approaches (Bing-Qiang et al., 2005).

Most of the tasks related to the navigation of a mobile robot can be formulated on the abstract level in terms of behavioral sequences. Nevertheless, the problem of automatic generation of a robot controller able to produce successful behavioral sequences with sufficient stability is still waiting for efficient solutions.

One of the few approaches to the solution of this problem is the Dynamic Field Theory (DFT) (Erlhagen and Schöner, 2002). Algorithms based on the DFT allow forming the "memory" of required action sequences and then efficiently reproduce them (Sandamirskaya and Schöner, 2010). The main disadvantage of the DFT is necessity of manual robot guiding through required sequences during learning phase. Such supervised learning makes it difficult to use this method in the tasks associated with autonomous exploration

of an environment with multiple alternative sequences. DFT algorithm also does not fully resolve the problem of autonomous detection of critical environment properties of the target state (Luciw et al., 2013).

At the same time one of the most promising approach in this area is neuroevolution. This approach has been applied from the pioneering works (Mondada and Floreano, 1995) until the most recent advances (Knudson and Tumer, 2011). Nowadays it is used for the complex behavior development in the navigational tasks including ones that require formation of a short-term memory, e.g. in T-maze task (Durr et al., 2008; Ollion et al., 2012).

To address the problem of adaptive behavior in multigoal environments we proposed neuroevolutionary algorithm with growing topology. In the work (Lakhman and Burtsev, 2013) this algorithm was studied in the context of neurocontroller evolution in abstract environment with many nested target action sequences. Here we apply this neuroevolutionary algorithm to the task of robot navigation.

The Model

Goal-directed behavior usually requires a sequence of effective actions to be completed successfully. In our study we simulate this with the following task for a mobile robot: robot must visit colored cubes in different target sequences. Visiting a cube of a certain color is interpreted as a single action and different goals correspond to different sequences of cubes visited. Robot is considered to have made an action in the case if its center of mass got into the circumference of a small radius around a corresponding cube.

Each target action sequence has its corresponding reward. The value of this reward is directly proportional to the number of cubes in it. When target sequence of actions is completed by an agent the corresponding reward is reset to zero and then linearly recovers to the initial level. Robot has no explicit access to the information about accumulated reward or moments of goals' achievement. Total reward accumulated by the robot throughout its presence in the arena affects its reproductive fitness in the evolutionary process:

$$V_{\text{total}} = \sum_{t} \sum_{i \in G(t)} v_i \times \min\left(\frac{t - t_i^{\text{last}}}{T^{\text{rec}}}, 1\right) , \quad (1)$$

where $G\left(t\right)$ is a set of goals reached at time $t,\,v_i$ is a reward associated with i^{th} goal, t_i^{last} is a time step when the agent has previously reached i^{th} goal, T^{rec} is a recovery time of goal's reward. It is also important to stress the fact that visiting a cube which is not present in the current target sequence leads to termination of that sequence and not receiving any reward for it.

For our experiments we simulated a small two wheeled E-PUCK robot (Mondada et al., 2009) with ENKI simulator (Magnenat et al., 2009). Robot's neurocontroller consists of formal neurons with truncated positive sigmoid activation function:

$$y = \begin{cases} \frac{1}{1 + e^{-2h}} & \text{if } h > 0\\ 0 & \text{otherwise} \end{cases}$$
 (2)

where h is a neuron's potential.

An architecture of neurocontroller is shown on the Fig. 1. Four input neurons receive values from the frontal proximity sensors that are inversely proportional to the distance to obstacle. Nine other input neurons encode the signal from camera. Environment in the simulation is two-dimensional so the robot's camera records a one-dimensional image with a viewing angle of 180 degrees. This image is then divided

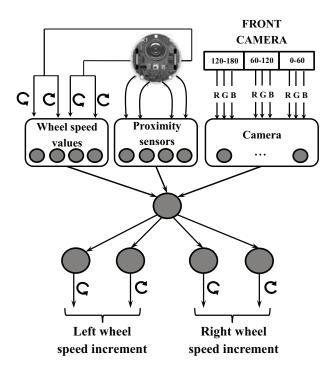


Figure 1: Architecture of the initial neurocontroller

into three sectors corresponding to the angle intervals of [0;60), [60;120] and (120-180] degrees. Values of each pixel in red, green and blue color channels for each sector are then summed and divided by the number of pixels in that sector, providing an average normalized value for R, G and B channels separately. Then these values are passed to the corresponding input neurons. The last four input neurons encode current rotation speed of both wheels separately for the different directions. If the wheel is spinning clockwise the signal is transferred to the first neuron of the pair, if it's spinning counterclockwise than it is transferred to the second one. Values of all inputs are normalized to fit a range from 0 to 1. We should note that robot has no access to its own absolute coordinates in a model environment.

Four output neurons of the neural network control wheels rotation. Each neuron encodes positive or negative increment of the speed of rotation of a particular wheel. Thereby the initial neural network controller of a robot consists of 17 input, 4 output and one interneuron.

In all simulations the first population consisted of robots controlled by neurocontrollers with the initial architecture (Fig. 1) but later in evolution neural networks grow in size. Evolution of a neural controller is implemented in accordance with the algorithm suggested in the study (Lakhman and Burtsev, 2013). This algorithm includes several types of topological mutations allowing to grow neural network: *neuron's duplication mutation*, addition and deletion of synapses. Neuron's duplication mutation implies that two

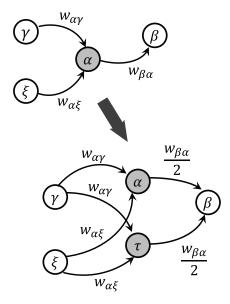


Figure 2: Schematic representation of the mutation operator neuron duplication. During duplication the "offspring" neuron τ inherits all synaptic weights from the "parent" neuron α . $w_{\alpha\beta}$ denotes a synaptic weight of the connection between neurons α and β .

"offspring" neurons are formed from one "parent" neuron. These "offspring" neurons inherit the same connectivity structure but the weights of the output synapses are divided by two. Thus, postsynaptic neurons receive exactly the same value of signal as they did before the duplication. Two neurons in the aggregate perform the former function, but later in evolution could evolve independently.

Whenever mutation add a synapse between two neurons from the same layer, that were not connected previously, the new intermediate layer for the postsynaptic neuron is being constructed. All neurons from the upper layers are being shifted by one layer. Therefore we preserve the same connectivity structure as before: recurrent synapses are still recurrent and forward are still forward.

See Appendix for mutations parameter values.

Experimental study

The simulated environment is a square arena with an edge length of 10 meters. Walls that bound the environment are black in order to minimize interference with the robot's camera that receives the RGB values of the colored cubes. Each cube has an edge length of 10 cm and its own unique color. The event of cube visiting is counted only if the center of robot crosses the circle with the radius of 15 cm around the cube.

There were four cubes in the first series of experiments – yellow, green, red and blue. Cubes were placed at the equal distances from the center of the arena thus forming a square with an edge length of 1 meter. Clustering of cubes at the center of the arena was chosen to minimize possible

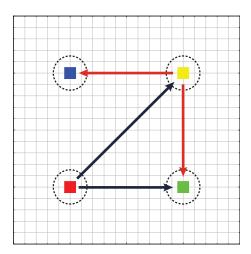


Figure 3: Topology of the "square" environment. Black arrows represent target sequences (goals) that are consist of two cubes, whereas red arrows represent the extension to the three-cube goals. Goal's hierarchy consists of four goals: *I*) Red – Yellow (RY); *II*) Red – Yellow – Green (RYG); *III*) Red – Yellow – Blue (RYB); *IV*) Red – Green (RG)

interactions of the robot with walls. The goal hierarchy in this environment has a tree structure with the goals depicted on the Fig. 3.

Each goal of the length two is associated with reward value of 20 and each goal consisting of three cubes gives reward 30. The agent's initial position and direction in the arena is random. The initial velocity is set to zero. Each robot is evaluated in the environment for 2 minutes and 40 seconds. After the agent achieves a goal, the reward for this goal drops to zero and then linearly recovers to the initial value in 10 seconds. Each population consists of 100 agents and evolution lasts for 3000 generations.

Dynamics of the average reward in the course of evolution for one of the runs is represented on the Fig. 4A (See Appendix for parameters of the evolutionary algorithm). The Fig. 4B illustrates a typical behavior of the agent of the 1000-th generation, just before the sharp rise of the average population fitness occurs. It's easy to notice that though the robot's track is already limited by a certain area near the cubes the robot hasn't stable behavioral strategy.

The behavioral cycle from the Fig. 4C is a stable goal attractor. Robot's trajectory converges to it from completely different starting points (Fig 5). It's important to notice that even in a situation when at the initial position the robot is facing the direction opposite to the cubes, it is able to efficiently find the central area and enter the same behavioral

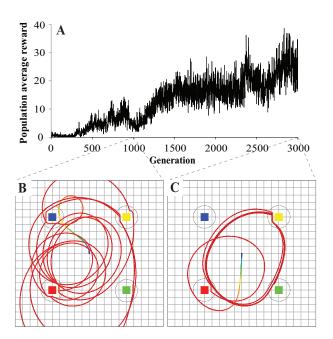


Figure 4: Evolution in the "square" environment. *A)* Typical dynamics of the average reward. *B)* Behavior of the best agent of 1000-th generation (note that the color of the track represents the robot's velocity). *C)* Typical behavior of an agent of 3000-th generation.

cycle independent of the initial direction.

After looking through the different behaviors of the agents with highest fitness we found that their trajectories usually had a circular form. This finding can be explained easily by symmetric location of cubes. To make the task more difficult we placed all cubes in one row retaining the same hierarchy of goals. Below we will refer to the first variant of the environment as "square" and the latter one as "linear".

We have extended the walls of the arena for the "linear" environment to the size 12 by 12 meters. This was done in order to provide enough distance between the outside cubes and the walls. All the other parameters of the simulation were the same.

The typical results for the evolutionary run in this experimental series and the example of the best agent's behavior are presented on the Fig. 6 and the Fig. 7A. There is notable increase of the average accumulated reward at about 1800-th generation indicating emergence of the effective behavior. Data shows that increase in fitness is accompanied by the growth in the number of interneurons and synapses in the agent's neurocontroller (Fig. 6B). The behaviors of successful robots in "linear" environment are more complex compared to the behaviors evolved in the "square" environment in terms of trajectory shape. The same behavior cycle consisting of target sequences (I)RY and (II)RYG now usually includes a small loop near the green cube. In most cases robots try to avoid the blue cube because of its low involvement in the target sequences. It is more "profitable" for the robot to drive up to yellow, red and green cubes in order to

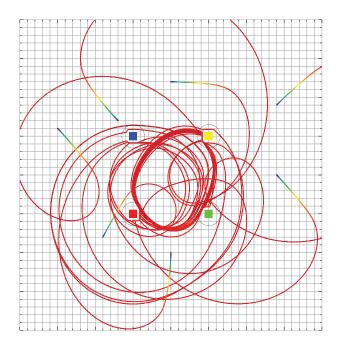


Figure 5: Convergence of the agent's behavior in the "square" environment from different starting points.

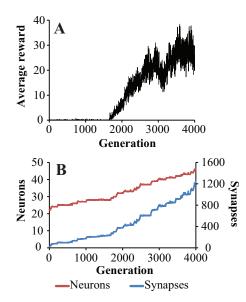


Figure 6: Evolution in the "linear" environment. *A)* Exemplar of the average reward dynamics. *B)* Dynamics of the population average of neurons and synapses.

get more reward because encounter with the blue cube can lead to the breakdown of the target action sequence and as a result in loss of both reward and time.

We run a series of experiments to clarify the degree to which robot's movement is determined by the colors of the cubes during its simulation run in the arena. We have chosen an agent with a stable behavioral policy for the goal (IV)RG. The idea for the experiment was to exchange colors of the green and blue cubes on the fly during lifetime when the robot had converged to the cycle. Observed results are depicted on the Fig. 7B–C. After the colors' swap the robot, which previously completed the goal (IV)RG, changed its trajectory and switch to another behavioral cycle, in which it achieved the sequence (I)RY. This switch could be explained by the presence of the cycle with the goal (I)RY in the previous generations.

As the next step we placed the robot evolved in the "linear" environment into the "square" environment (Fig. 8). Previously this robot produced behavior shown on the Fig. 7A. The behavioral cycle has been shortened and robot no longer visits green cube and hence does not achieve the long goal (II)RYG. Despite that fact its behavior remains quite efficient – it is able to achieve the goal (I)RY even though red and yellow cubes are much closer to each other in the "square" arena than they are in the "linear" one.

On the next stage of the study a population of agents evolved in the randomized "linear" arena to create less predictable environment. For every generation vertical position of each cube was set randomly. After 6000 generations 5 most successful neurocontrollers were chosen as founders

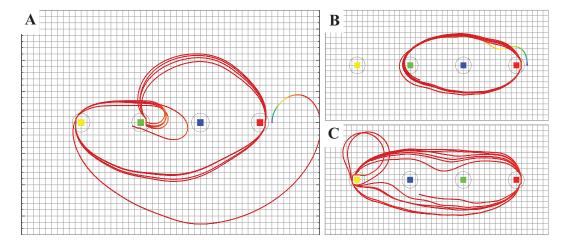


Figure 7: Behavior in the "linear" environment. A) Typical behavior of an agent from the last generation. B) RG-cycle before the swap of green and blue cubes (see text for details). C) RY-cycle after the swap of cubes (see text for details)

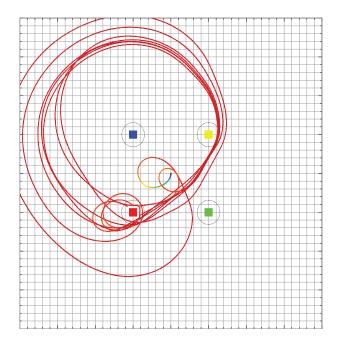


Figure 8: Behavior of an agent evolved in the "linear" environment during the run in the "square" environment

for the initial population set to evolve in the environment where location of cubes were distributed randomly vertically and horizontally in a rectangular area with size of 5 by 3 meters. The agent's life within the arena started at the randomly selected location in the same area. Goal hierarchy and all the geometrical parameters related to the cubes were the same.

One of the behavioral strategies evolved in the randomized environment is shown on the Fig. 9. The agent is able to adapt successfully to the different spatial configurations of cubes. It's behavior converges to the familiar cycle con-

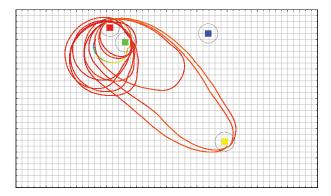


Figure 9: Example of the behavior evolved in the randomized environment.

sisting of goals (I)RY and (II)RYG in cases where red, yellow and green cubes are located close to each other as well as when they are quite distant. There is still a tendency to avoid the blue cube because of its low reward priority.

We also compared performance of our neuroevolutionary algorithm with the growing topology to the performance of evolution of fixed topology network. To conduct this experiment we built a model of initial network which had the same input and output layers setup but three fixed fully-connected feedforward interneuron layers consisting of 10, 8 and 6 neurons respectively. Thus the neurocontroller has 24 interneurons which is nearly the average number of interneurons at the end of evolution in the case of neuron duplications algorithm (as shown on the Fig. 6B). Such a setup makes a search space for the fixed topology algorithm comparable in complexity to the neuron duplications algorithm. Fig. 10 shows the results of comparison between the evolutionary algorithm with neuron duplications and a fixed net-

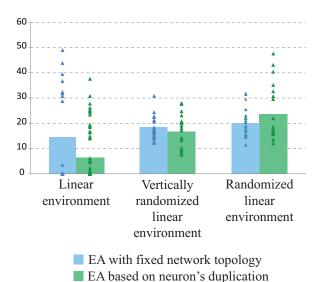


Figure 10: Comparison of evolutionary algorithms with fixed and growing network topology in the "linear" environment with different randomization degrees. There were 20 runs for each algorithm in each environment type (except for the standard linear environment - the neuron duplication algorithm was run 60 times). Columns represent rewards averaged over all runs and triangles represent the average reward values at the end of each run.

work topology algorithm.

On the Fig. 10 one can see that the average reward tend to rise with the task complexity in case of evolution with neuron duplications though in case of fixed network topology the increase is not so significant. This idea is confirmed by the reward distribution - we see that the difference in maximum reward tends to rise with the task complexity as well.

To make possible definition of more complex target sequences of actions a number of cubes in the environment was increased up to the seven. Each cube had fixed position at the same distance from the center of the arena. We called this environment "circular". Seven goals were defined as shown on the Fig. 11A. Typical behavioral cycles after evolution in the "circular" arena were circular (like the one on the Fig. 11B) but we also found more complex trajectories like the one depicted on the Fig. 12.

Best behavioral cycles evolved in the "linear" and "square" environments were formed by combination of the two-cube goal with its extension by one additional cube. However, best cycles for the "circular" problem consisted of two completely different goals (Fig. 12.). In this cycle agent achieves goal number IV (PURPLE-GREEN-AZURE) and then the number V (YELLOW-WHITE). It's important to notice that the evolutionary emergence of such behaviors had very low probability. Visiting a yellow cube after an

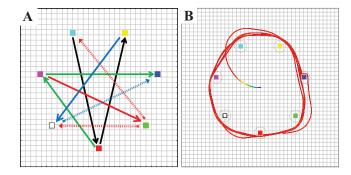


Figure 11: "Circular" environment. *A)* Goal's hierarchy in the "circular" environment (striped arrows represent the extended goal sequences). *B)* Circular behavioral cycle. Robot clearly avoids cubes that are not presented in the goal.

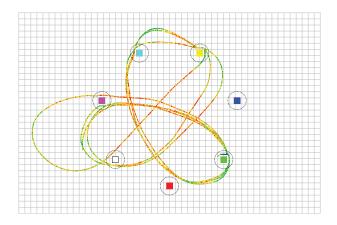


Figure 12: Behavior combining two goals in the "circular" environment (see text for details).

azure one is not enough for the goal to be completed because the agent has to visit the white cube afterwards without getting close to any other cube.

Conclusion

Simulation results demonstrate that neuroevolutionary algorithm with neuron duplication can be successfully applied not only in the abstract settings (Lakhman and Burtsev, 2013) but also to the problem of robot navigation in the multi-goal environment.

Robots driven by evolved neurocontrollers demonstrate robust behavior that makes possible completion of target action sequences independent of an initial robot position and direction. Experiments with randomized environments show that resulting neural nets is able to abstract the task of following particular sequence of cubes from the spatial location of cubes themselves.

The advantage of the neuroevolutionary algorithm with neuron duplication is proportional to the complexity of the task compared to the fixed network topology algorithm.

Acknowledgements

This work was partially supported by RFBR grant #13-04-01273.

Appendix

Evolutionary algorithm was run with the following parameters: population size -100 agents; probability of the synaptic weight mutation -0.6 (for each synapse); variance of the synaptic weight mutation -0.08; probability of adding a synapse -0.1 (for the whole network); probability of deleting a synapse -0.05 (for the whole network); probability of neuron's duplication -0.007 (for the whole network).

Environment specific parameters: simulation step -0.1 seconds.

"Square" and "linear" environments: reward recovery time – 10 seconds; single agent simulated lifetime – 160 seconds; number of generations – 4000.

Randomized "linear"-like environment: reward recovery time – 10 seconds; single agent simulated lifetime – 160 seconds; number of generations – 6000.

"Circular" environment: reward recovery time – 120 seconds; single agent simulated lifetime – 320 seconds; number of generations – 9000.

References

- Bing-Qiang, H., Guang-Yi, C., and Min, G. (2005). Reinforcement learning neural network to the problem of autonomous mobile robot obstacle avoidance. In *Proceedings of the International Conference on Machine Learning and Cybernetics*, volume 1, pages 85–89.
- Cassandra, A., Kaelbling, L., and Kurien, J. (1996). Acting under uncertainty: discrete bayesian models for mobile-robot navigation. In *Proceedings of the IEEE/RSJ International Con*ference on Intelligent Robots and Systems, volume 2 of IROS '96, pages 963–972.
- Durr, P., Mattiussi, C., Soltoggio, A., and Floreano, D. (2008). Evolvability of neuromodulated learning for robots. In *Proceedings of the ECSIS Symposium on Learning and Adaptive Behaviors for Robotic Systems*, LAB-RS '08, pages 41–46.
- Erlhagen, W. and Schöner, G. (2002). Dynamic field theory of movement preparation. *Psychological Review*, 109(3):545– 572.
- Janglova, D. (2004). Neural networks in mobile robot motion. *International Journal of Advanced Robotic Systems*, 1:15–22.
- Knudson, M. and Tumer, K. (2011). Adaptive navigation for autonomous robots. *Robotics and Autonomous Systems*, 59(6):410–420.
- Lakhman, K. and Burtsev, M. (2013). Neuroevolution results in emergence of short-term memory in multi-goal environment. In *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, GECCO '13, pages 703–710, New York, NY, USA. ACM.

- Luciw, M., Lakhman, K., Richter, M., Kazerounian, S., and Sandamirskaya, Y. (2013). Learning the perceptual conditions of satisfaction of elementary behaviors. In *Robotics Science and Systems (RSS) Workshop on Active Learning in Robotics: Exploration, Curiosity, and Interaction.*
- Magnenat, S., Waibel, M., and Beyeler, A. (2009). Enki an open source fast 2d robot simulator, http://home.gna.org/enki/.
- Martinez-Cantin, R., Freitas, N., Brochu, E., Castellanos, J., and Doucet, A. (2009). A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots*, 27(2):93–103.
- Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C., Floreano, D., and Martinoli, A. (2009). The e-puck, a robot designed for education in engineering. In Gonalves, P., Torres, P., and Alves, C., editors, *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, volume 1, pages 59–65, Portugal. IPCB: Instituto Politcnico de Castelo Branco.
- Mondada, F. and Floreano, D. (1995). Evolution of neural control structures: some experiments on mobile robots. *Robotics and Autonomous Systems*, 16(2-4):183–195. Moving the Frontiers between Robotics and Biology.
- Ollion, C., Pinville, T., and Doncieux, S. (2012). Emergence of memory in neuroevolution: Impact of selection pressures. In Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO '12, pages 369–372, New York, NY, USA. ACM.
- Sandamirskaya, Y. and Schöner, G. (2010). An embodied account of serial order: How instabilities drive sequence generation. *Neural Networks*, 23(10):1164–1179.
- Smart, W. and Kaelbling, L. (2002). Effective reinforcement learning for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 4 of *ICRA* '02, pages 3404–3410.

Twitter as Social Sensor: Dynamics and Structure in Major Sporting Events

Yuki Takeichi, Kazutoshi Sasahara, Reiji Suzuki and Takaya Arita

Graduate School of Information Science, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan {tacke, sasahara}@nagoya-u.jp

Abstract

Twitter often behaves like a "social sensor" in which users actively sense real-world events and spontaneously mention these events in cyberspace. Here, we study the temporal dynamics and structural properties of Twitter as a social sensor in major sporting events. By examining Japanese professional baseball games, we found that Twitter as a social sensor can immediately show reactions to positive and negative events by a burst of tweets, but only positive events induce a burst of retweets to follow. In addition, retweet networks during the baseball games exhibit clear polarization in user clusters depending on baseball teams, as well as a scale-free in-degree distribution. These empirical findings provide mechanistic insights into the emergence and evolution of social sensors.

Introduction

Online social media sites, such as Twitter and Facebook, have become increasingly popular, to the point that they are now essential tools in everyday life, thereby facilitating massive, near-realtime, networked social interactions in cyberspace. In addition, these media can have an impact not only in cyberspace but also in the physical-world. For example, it was reported that Twitter helped Arab Spring activists to spread and share information, playing a key role in the ensuing revolutionary social movements¹. Thus, online social media can work as interfaces between cyberspace and real-world environments, connecting people and information in some nontrivial ways. Consequently, online social media form a hybrid system of users and the web, which may behave like a single organism that evolves in time, providing a new research subject for the study of artificial life.

Many social media studies have already been conducted, though not in the context of artificial life. Focusing particularly on Twitter, we see that previous studies have reported its unique characteristics, such as the structural properties of user networks (Kwak et al., 2010; Bollen et al., 2011a), the nature of social interactions (Grabowicz et al., 2012; Conover et al., 2012) and information diffusion

(Romero et al., 2011; Weng et al., 2012), collective attention (Lehmann et al., 2012; Sasahara et al., 2013) and collective mood (Golder and Macy, 2011; Dodds et al., 2011), and users' dynamics related to particular real-life events (Sakaki et al., 2010; Borge-Holthoefer et al., 2011; González-Bailón et al., 2011). Twitter data were also used to detect emerging topics (Takahashi et al., 2014) and to predict the stock markets (Bollen et al., 2011b).

This paper focuses on Twitter as a "social sensor," a new type of emergent collective behavior in the social age. Twitter allows users to read, post, and forward a short text message of 140 characters or less, called "tweets," in online user networks. As shown in Fig. 1, Twitter users actively sense real-world events and spontaneously make utterances about these events by posting tweets, which immediately spread over online user networks. In addition, such information cascades can be amplified by chains of "retweets" (forwarded tweets) from other users, called followers. This is not a passive one-shot process, but rather an active process that is recurrently happening and constantly evolving due to changes both in the physical-world and in cyberspace. Consequently, the Twitter system can behave like a social sensor, exhibiting collective dynamics and a distinct structure linked with target events. This is true in principle, and the previous studies mentioned above have revealed some aspects of social sensors. However, little is known about the dynamic nature of social sensors which cannot be explained solely by "bursts of tweets."

We therefore conducted a case study of Twitter as a dynamic social sensor in major sporting events—Japan's 2013 Nippon Professional Baseball (NPB) games—by focusing on co-occurrences of tweets and retweets. These target events were suitable for our primary study because it is known that major sporting events are the subjects of strong collective attention of viewers, which gives rise to a large volume of tweets and retweets (Bagrow et al., 2011; Sasahara et al., 2013). Our study provides key insights into how and when the collective dynamics of social media users emerge and function as a social sensor.

http://www.arabmediasociety.com/?article= 816

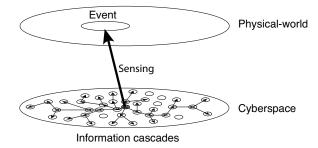


Figure 1: Schematic illustration of a social sensor. Nodes in cyberspace represent Twitter users. The thick arrow represents a user sensing a real-world event, and thin arrows represent the corresponding information cascades by means of tweet and retweet.

Methods

For this case study, we collected a comprehensive dataset by using hashtags, which are used to categorize tweets by keywords. The dataset was analyzed to explore the dynamics and structure of Twitter as a social sensor linked with major sporting events, described as follows.

Tweet Collection

We collected tweets surrounding 19 baseball games from the Climax Series (the annual playoff series in NPB) held from October 12 to 21, 2013, and from the Japan Series (the annual championship series in NPB) held from October 26 to November 3, 2013 in NPB. To this end, we selected hashtags related to Japanese professional baseball, such as #giants and #rakuteneagles, each of which represents the name of a professional baseball team by reference to a hashtag cloud site². Then, we continuously crawled tweets with these target hashtags by using Twitter Search API³. This crawling resulted in 528,501 tweets for the baseball games in total. Each piece of tweet data contains a text message with at least one hashtag and the metadata, including the timestamp and the user profile.

Measurement of Temporal Correlation between Tweets and Retweets

The burst-like increases of tweets are often followed by those of retweets, which recurrently occur especially when positive events happen in the physical-world, as we will see later on. To measure temporal correlation between tweet and retweet count time series, we measured a cross-correlation function defined as follows (Venables and Ripley, 2002):

$$R_k(i,j) = \frac{C_k(i,j)}{\sqrt{C_0(i,i)C_0(j,j)}},$$

where

$$C_k(i,j) = Cov(y_n(i), y_{n-k}(j))$$

= $E[(y_n(i) - \bar{y}_n(i))(y_{n-k}(j) - \bar{y}_{n-k}(j))].$

 $R_k(i,j)$ varies between -1 and 1. In our analysis, $y_n(i)$ and $y_n(j)$ are the time series of tweets and retweets counted by 10 sec, respectively, and k is a time lag of $y_n(j)$ to $y_n(i)$. We changed k between 0 and 5 min at 10 sec intervals, because the following retweets always occur after the bursts of tweets. We adopted the maximum value of $R_k(i,j)$ as a measure of temporal correlation between tweet and retweet time series (denoted by R_{max}).

Construction of Retweet Networks

The structures of social sensors linked with major sporting events are examined using complex networks. Complex networks consist of a large number of nodes with sparse connections between them, and they are used to describe, analyze, and model real-world networks, ranging from biological systems to social systems to artificial systems (Newman, 2010).

Using official retweets (not user retweets—posts with "RT" by hand), we constructed "retweet networks," in which each node represents a user and a directed edge is attached from user B to user A if user B retweets a tweet posted originally by user A. Note that if another user C retweets a user B's retweet, a directed edge is connected from user C to user A (i.e., tweet origin). This is due to the official retweet specification of the Twitter system. Thus, influential users (also known as "hub" users) whose tweets are preferentially retweeted are represented as nodes with many incoming edges.

The resulting retweet networks are visualized in a forcedirected layout algorithm called OpenOrd⁴ using Gephi⁵. The size of nodes is proportional to the logarithm of the number of in-degrees. In addition, cumulative in-degree distributions are calculated from retweet networks to access their structural properties.

Results

First, we show an example of the tweet and retweet dynamics of a baseball game in the 2013 Japan Series. Then, we look into the temporal correlations of tweets and retweets as a unique feature of dynamic social sensors in 19 baseball games from the Climax Series and the Japan Series. Finally, we examine Twitter as a social sensor in terms of user interactions by constructing and analyzing retweet networks.

²http://hashtagcloud.net

https://dev.twitter.com/docs/api/1.1/get/ search/tweets

https://marketplace.gephi.org/plugin/ openord-layout/

⁵https://gephi.org

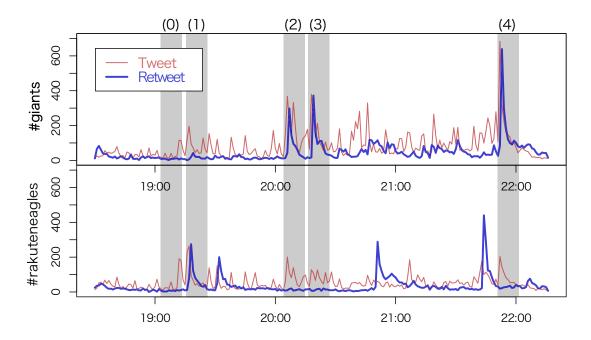


Figure 2: Example of tweet and retweet time series for the sixth round in the 2013 Japan Series in NPB. Red lines denote tweets and blue lines denote retweets. The upper panel shows tweets for the Giants (#giants) and the lower panel for the Eagles (#rakuteneagles). There is no event in (0). See the text for (1)-(4).

Dynamics of Tweets and Retweets

Figure 2 shows tweet and retweet time series for the 6th round in the 2013 Japan Series. In this game, the Yomiuri Giants beat the Tohoku Rakuten Golden Eagles by a score of 4-2. We see some co-occurrences of burst-like increases in tweets and retweets in Fig. 2 (1)-(4), each of which corresponds to the following events, respectively:

- (1) The Eagles batted around in the bottom of the second inning, scoring two runs.
- (2) The Giants turned the game around in the top of the fifth inning, scoring three runs.
- (3) The Giants added another run in the top of the sixth inning.
- (4) The Giants won the game.

In Fig. 2, the co-occurring bursts of tweets and retweets more frequently emerged in the context of the Giants (the winning team) than the Eagles (the losing team). Figure 3 exemplifies that during event (3), positive tweets such as "Oh goody!" and "go-ahead homer!" were posted with #giants, whereas negative tweets such as "Oh, no!" and "strike out ..." were posted with #rakuteneagles. Thus, once a particular event happens during a baseball game, users spontaneously post a scream of delight from

the winning side and one of disappointment from the losing side. In other words, without such events, there is no strong bias against a tweet's polarity, positive or negative, as seen in Fig. 3 (0). These findings suggest that in baseball games, Twitter as a social sensor can immediately show reactions to positive and negative events by a burst of tweets, but only positive events induce a burst of retweets to follow.

On the basis of these observations, we assume that a temporal correlation between tweet and retweet time series would work as a measure of collective positive reactions of users in baseball games, which we will quantitatively examine in the next section.

Co-occurring Bursts of Tweets and Retweets

We now turn to the co-occurring bursts of tweets and retweets as a social sensor measure in major sporting events. To this end, we computed and compared a cross-correlation function for tweet and retweet time series, defined in the Methods section, from the Japan Series (seven games) and the Climax Series for the Central League (five games) and for the Pacific Leagues (seven games). We examined 19 games in total. We limited our analysis from the start time to one hour post-game for each game.

Figure 4 (left) shows the maximum values of the cross-correlation function (R_{max}) of tweet and retweet time series for the Giants (G) and the Eagles (E) across seven games in the 2013 Japan Series. In this figure, we can confirm that the

winning team has R_{max} greater than that of the losing team in all games. Moreover, we found two interesting features: in the first round, R_{max} for the Giants was much larger than that of the Eagles, because this was a one-sided game and the Giants went on to win with consummate ease; in the fifth round, both teams showed an equivalent R_{max} value, because it was a close game. These results seem reasonable because, as mentioned above, a greater R_{max} value is associated with simultaneous bursts of tweets and retweets, which are in turn associated with significant game events, such as a base hit or home run. Therefore, the deviations of R_{max} are attributed to the degree of excitement of a game, which corresponds to significant scoring events.

We examined whether this notable property of a dynamic social sensor holds for other baseball games in the 2013 Climax Series. R_{max} worked as a good measure of positive reactions in the social sensor in 16 of 19 games. As shown in Fig. 4 (middle and right panels), this property holds true except in the case of three games: the second round in the Central League Climax Series and the fifth and seventh rounds in the Pacific League Climax Series. Two of these exceptions were based on the non-stationarity of tweeted and retweeted time series, in which fans generated a single sustained burst of retweets regarding a winning run after a long pitchers' duel. The other exception was based on inactive retweet reactions; for some reason, fans were not well excited or focused. Thus, R_{max} cannot be applied to both cases, which is a potential disadvantage of this measure.

We next classified the computed R_{max} values into two groups, one is the winning team group and the other the losing team group, and compared their means statistically. The result shows a significant difference between the two groups (t-test, P < 0.05), as shown in Fig. 5, meaning that greater R_{max} values are related to winning games. Our speculation described in the previous section has now been statistically confirmed. Therefore, we conclude that the positive collective reactions of a social sensor, measured by R_{max} , are highly indicative of winning in baseball games, and probably in other professional team sports, such as football and basketball.

Retweet Networks and Social Interactions

To quantify the structure of Twitter as a social sensor, we constructed and analyzed retweet networks related to the sixth round in the 2013 Japan Series using tweet data with #giants and #rakuteneagles. As mentioned before, nodes represent users and directed links represent official retweets between them, and colors correspond to hashtags.

The retweet network (A) corresponds to event (1) where the Eagles got two runs in the second inning, and the network (B) corresponds to event (2) and (3) where the Giants turned the game around. These networks have distinct structural features. First, the retweet networks (A) and (B)

	Fig. 2 (0)	Fig. 2 (3)
#giants	菅野頼みだのう・・・#giants 完璧に抑えられてるの。うう。#giants 川・・・)ツー・今日の巨人打線はマー君相手 巨人ファンです!試合中このアカウントでつぶ パット折られすぎやん、#giants 打ててないな・・・・(ゲ・ /)/心配である。 いやでも絶対諦めん。#giants 150 を超えるストレート、キレッキレッの変 今、坂本選手を打ち取ったのはツーシームか。 今日の試合は 先制点をとられたらそのチーム 菅野がんばれ!!! #giants #kyojin 巨人の攻撃終わるの早すざるんだよなあ・・ まったく負けないって気持ち悪いぜ #giants	巨人逆転 #giants やったー!!!!!!!!!!!# giants けー (∀) —!! #giants 逆転きたああああああああああああああああああああああああああああああああああああ
#rakuteneagles	先制点早く欲しい。#rakuteneagles #t #rakuteneagles #t #v	まじか逆転 #rakuteneagles のおおおおおおおおおおおおおおおおおおおおおおおおおおおおおおおおおおおお

Figure 3: Examples of tweets' contents by #giants and #rakuteneagles. Red texts denote positive exclamations and blue ones denote negative exclamations posted by users. Without any particular events in the game (Fig. 2 (0)), there are not many positive or negative tweets from either hashtag. With an event (Fig. 2 (3)), there are many positive tweets with #giants and in contrast negative tweets with #rakuteneagles.

are composed of two main sub-networks, one is a cluster of the Giants fans (green) and the other is a cluster of the Eagles fans (blue). Within the same sub-networks there are numerous retweet interactions; however, between different sub-networks (i.e., between the Giants cluster and the Eagles cluster) there are fewer retweet interactions. A similar network topology was previously reported for the retweet networks of online political activity (Conover et al., 2012). Second, the Giants cluster involves several hub users (large nodes) who are preferentially retweeted, whereas there was a single hub user in the Eagles cluster. It turns out that these hub users are either the official account for the Giants and the Eagles or enthusiastic fans. Interestingly, there are a few retweets with both hashtags.

The bottom panels in Fig. 6 show the in-degree distributions (double logarithmic plots) of the corresponding retweet networks, respectively. These in-degree distributions exhibit a scale-free property; a long tail proofs the existence of hub users' many retweeted posts, although we have visually confirmed this above. Furthermore, it should be noted that the tails of the in-degree distributions tend to shift to the right (i.e., greater k) on the winning side. In network (A), the tail is much longer in the Eagles cluster than the Giants cluster, while in network (B) the situation is opposite. Although we used tweets with #giants and #rakuteneagles for analysis, a bipolar structure as well as a scale-free nature we have observed are not at all trivial, but rather provides hints

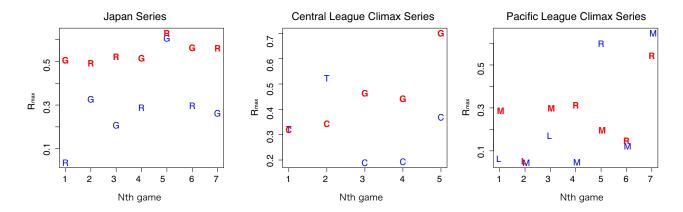


Figure 4: Cross-correlation values (R_{max}) between tweet and retweet time series for the 2013 Japan Series (left) and the 2013 Climax Series for the Central League (middle) and the Pacific League (right). Red denotes the winning term and blue denotes the losing team. G: Yomiuri Giants, R: Tohoku Rakuten Golden Eagles, T: Hanshin Tigers, C: Hiroshima Toyo Carp, M: Chiba Lotte Marines, L: Saitama Seibu Lions.

on how users interact, behaving like a social sensor.

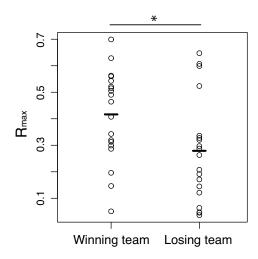


Figure 5: Distributions of R_{max} in the winning team group and the losing team group, with the mean values (crossbars).

Discussion

In this paper, we have demonstrated the temporal dynamics and structural properties of Twitter in major sporting events. Our results provide empirical evidence of how and when collective dynamics of users in the web function as a social sensor. We found that co-occurring bursts of tweets and retweets happen frequently and recurrently in winning teams in baseball games, and consequently R_{\max} for these time series can be a good indicator of winning or losing the game.

This notable property, however, is not necessarily true for other sporting events, and no doubt it depends on the type of sport. This is because different sporting events may have different "affordance" (Gibson, 1977), thereby possibly inducing distinct yet coherent user reactions. For example, in two-team sports such as baseball and football, there are detailed rules with a scoring mechanism that can prompt fans to be more aware of a game's progress. This situation elicits spontaneous, polarized tweet and retweet reactions to scoring events among fans of different teams. In contrast, in multi-team sports such as car racing, rules are simple and there is no scoring mechanism, which may deprive fans of a chance to react the progress of a race. In this situation, tweet and retweet reactions can occur in a different fashion than with two-team sports. In fact, we observed such a case in the 2013 F1 Japanese Grand Prix (data not shown). Nevertheless, we think that the measurement of co-occurring bursts of tweets and retweets using R_{max} can be applied to a wider class of major sporting events, and probably other social events as well. Comparing social sensors in different types of events is thus important for the fundamental understanding of a new type of emergent collective behavior.

Furthermore, we revealed that the retweet networks for the baseball game exhibit a scale-free property, with hub users or influentials who contribute to cascades of retweets, as with other retweet networks for meme diffusion (Weng et al., 2012) and collective attention (Sasahara et al., 2013). In addition, the retweet networks for the baseball game had bipolar sub-network structures depending on the baseball teams, as with retweet networks for online political activity (Conover et al., 2012), indicating the possibility of the same underlying design principle. To assess the generality of our findings, further investigations are necessary using a wide variety of major sporting events and other types of social events across different social media.

In summary, Twitter is a "social sensor" in that it allows

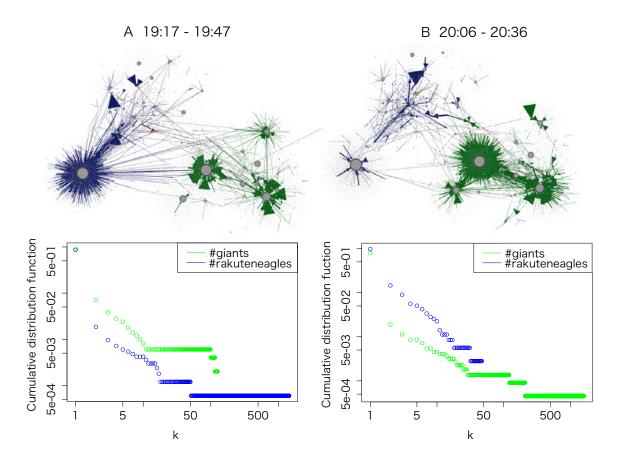


Figure 6: Retweet networks and their degree distributions in the sixth round of the 2013 Japan Series. The retweet network (A) consists of tweets generated during 30 minutes from 19:17, in which more retweets were generated with #rakuteneagles. The retweet network (B) consists of tweets generated during 30 minutes from 20:16, in which more retweets were generated with #giants. Green lines and circles denote #giants and blue lines and circles denote #rakuteneagles.

users to immediately and collectively react real-time events by tweeting and it is "active" in that users selectively retweet favorite posts, resulting in the simultaneous bursts of tweets and retweets that spread over polarized scale-free user networks. Our results offer mechanistic insights into the emergence and evolution of a dynamic social sensor. Gaining this insight is critical not only for a better understanding of the social web as a decentralized, independent, uncontrollable, living system but also for developing methods of living technology (Bedau et al., 2010) for future web-based systems. In addition, the accumulation of case studies of this kind is fundamental to artificial life study to understand a new type of complexity that arises from a collective human nature on the web.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 24700291. KS was supported by The Hori Science and

Arts Foundation Research Grant.

References

Bagrow, J. P., Wang, D., and Barabási, A.-L. (2011). Collective Response of Human Populations to Large-Scale Emergencies. PLoS ONE, 6(3):e17680.

Bedau, M. A., McCaskill, J. S., Packard, N. H., and Rasmussen, S. (2010). Living Technology: Exploiting Life's Principles in Technology. *Artificial Life*, 16(1):89–97.

Bollen, J., Goncalves, B., Ruan, G., and Mao, H. (2011a). Happiness is Assortative in Online Social Networks. *Artificial Life*, 17(3):237–251.

Bollen, J., Mao, H., and Zeng, X. (2011b). Twitter Mood Predicts the Stock Market. *Journal of Computational Science*, 2(1):1–8.

Borge-Holthoefer, J., Rivero, A., García, I., Cauhé, E., Ferrer, A., Ferrer, D., Francos, D., Iñiguez, D., Pérez, M. P., Ruiz, G., Sanz, F., Serrano, F., Viñas, C., Tarancón, A., and Moreno, Y.

- (2011). Structural and Dynamical Patterns on Online Social Networks: The Spanish May 15th Movement as a Case Study. *PLoS ONE*, 6(8):e23883.
- Conover, M. D., Gonçalves, B., Flammini, A., and Menczer, F. (2012). Partisan Asymmetries in Online Political Activity. *EPJ Data Science*, 1(1):6.
- Dodds, P. S., Harris, K. D., Kloumann, I. M., Bliss, C. A., and Danforth, C. M. (2011). Temporal Patterns of Happiness and Information in a Global Social Network: Hedonometrics and Twitter. *PLoS ONE*, 6(12):e26752.
- Gibson, J. J. (1977). *The Theory of Affordances*, pages 127–143. Perceiving, Acting, and Knowing: Towards an Ecological Psychology. Hoboken, NJ: John Wiley & Sons Inc.
- Golder, S. A. and Macy, M. W. (2011). Diurnal and Seasonal Mood Vary with Work, Sleep, and Daylength Across Diverse Cultures. *Science*, 333(6051):1878–1881.
- González-Bailón, S., Borge-Holthoefer, J., Rivero, A., and Moreno, Y. (2011). The Dynamics of Protest Recruitment through an Online Network. *Scientific Reports*, 1:197.
- Grabowicz, P. A., Ramasco, J. J., Moro, E., Pujol, J. M., and Eguíluz, V. M. (2012). Social Features of Online Networks: The Strength of Intermediary Ties in Online Social Media. *PLoS ONE*, 7(1):e29358.
- Kwak, H., Lee, C., Park, H., and Moon, S. (2010). What is Twitter, a Social Network or a News Media? In *Proceedings of the 19th International Conference on World Wide Web*, pages 591–600.
- Lehmann, J., Gonçalves, B., Ramasco, J. J., and Cattuto, C. (2012). Dynamical Classes of Collective Attention in Twitter. In *Proceedings of the 21st International Conference on World Wide Web*, pages 251–260.
- Newman, M. E. (2010). Networks: An Introduction. Oxford University Press.
- Romero, D. M., Meeder, B., and Kleinberg, J. (2011). Differences in the Mechanics of Information Diffusion Across Topics: Idioms, Political Hashtags, and Complex Contagion on Twitter. In *Proceedings of the 20th International Conference on World Wide Web*, pages 695–704.
- Sakaki, T., Okazaki, M., and Matsuo, Y. (2010). Earthquake Shakes Twitter Users: Real-Time Event Detection by Social Sensors. In *Proceedings of the 19th International Conference* on World Wide Web, pages 851–860.
- Sasahara, K., Hirata, Y., Toyoda, M., Kitsuregawa, M., and Aihara, K. (2013). Quantifying Collective Attention from Tweet Stream. *PLoS ONE*, 8(4):e61823.
- Takahashi, T., Tomioka, R., and Yamanishi, K. (2014). Discovering Emerging Topics in Social Streams via Link-Anomaly Detection. Knowledge and Data Engineering, IEEE Transactions on, 26(1):120–130.
- Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S.* Springer-Verlag, 4th ed. edition.
- Weng, L., Flammini, A., Vespignani, A., and Menczer, F. (2012). Competition Among Memes in a World With Limited Attention. *Scientific Reports*, 2:335.

Hybrid Control for Large Swarms of Aquatic Drones

Miguel Duarte, Sancho Moura Oliveira and Anders Lyhne Christensen

Instituto de Telecomunicações &
Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal
{miguel_duarte, sancho.oliveira, anders.christensen}@iscte.pt

Abstract

Maritime tasks, such as surveillance and patrolling, aquaculture inspection, and wildlife monitoring, typically require large operational crews and expensive equipment. Only recently have unmanned vehicles started to be used for such missions. These vehicles, however, tend to be expensive and have limited coverage, which prevents large-scale deployment. In this paper, we propose a scalable robotics system based on swarms of small and inexpensive aquatic drones. We take advantage of bio-inspired artificial evolution techniques in order to synthesize scalable and robust collective behaviors for the drones. The behaviors are then combined hierarchically with preprogrammed control in an engineeredcentric approach, allowing the overall behavior for a particular mission to be quickly configured and tested in simulation before the aquatic drones are deployed. We demonstrate the scalability of our hybrid approach by successfully deploying up to 1,000 simulated drones to patrol a 20 km long strip for 24 hours.

Introduction

Coastal countries have faced an increased spending over the years in order to carry out maritime tasks. In Italy, for instance, the problem of illegal immigration (Monzini, 2007; Carling, 2007) and organized crime (Lutterbeck, 2006) has contributed for the growth of the Guardia di Finanza's budget from \$1.10B to \$3.21B, during the 1990s, with an increase both in personnel (28%) and equipment, which, in 1999, counted 582 boats (78% increase), 96 helicopters (41% increase), and a total of 14 airplanes (Lutterbeck, 2004). In Spain, immigrants crossing the Gibraltar Strait through Morocco have lead the Spanish government to implement the Sistema Integrado de Vigilancia Exterior (SIVE), which is composed of military technology such as fixed and mobile radars, infrared sensors, and traditional aquatic and aerial vehicles (Lutterbeck, 2006).

Since current state-of-the-art systems require a large human crew in order to successfully execute maritime tasks, efforts have been made to adapt unmanned vehicle technology, such as *unmanned aerial vehicles* (UAV) and *unmanned surface vehicles* (USV) for these tasks. However, current UAV

and USV systems tend to be expensive to acquire and operate, and therefore tend to be comprised of only a small number of units (Schwing, 2007). Examples are the military-grade General Atomics MQ-9 Reaper UAV, and the Protector USV (Yan et al., 2010). Only recently have technological advances made it possible (and affordable) for researchers to begin to study the use of medium or large-sized swarms of inexpensive and unmanned autonomous vehicles (Manley, 2008).

In this paper, we propose a system composed of a swarm of relatively simple aquatic drones with decentralized control for maritime missions. Our proposed system is composed of potentially hundreds or thousands of small autonomous aquatic drones. Such decentralized systems present numerous potential benefits compared with monolithic robotic systems (Bayindir and Şahin, 2007). On the one hand, the use of multiple drones introduces redundancy in the system, which reduces the impact of hardware failures and has the potential to improve operational efficiency by allowing a larger area to be covered simultaneously. On the other hand, automating such missions considerably reduces the maintenance cost (smaller human crew) while increasing availability and scalability.

We employ our novel hybrid control approach (Duarte et al., 2014b) that combines evolutionary robotics (ER) techniques and preprogrammed logic to facilitate configurable, decentralized control. We evolve several simple selforganized behaviors, which would be challenging to manually program. The use of ER techniques enables the synthesis of robust and scalable controllers for simple tasks, which are then combined using high-level preprogrammed logic for the complete task. In this way, our approach addresses fundamental challenges in the synthesis and use of self-organized control for real-world swarm systems. By dividing control into individual components, (i) it becomes computationally feasible to evolve self-organized control for large-scale systems, (ii) we diminish bootstrapping issues since solutions to a set of simpler tasks are sought instead of solutions to the more complex, global task, and (iii) control for new missions can be composed based on existing behavioral components.

Our study is novel in three respects: (i) we apply evolutionary techniques to the domain of large swarms of aquatic drones, a combination which to the best of our knowledge is unexplored, (ii) we demonstrate our hybrid control approach applied to a multirobot system with up to 1,000 drones, where it has previously been applied only to single-robot systems (Duarte et al., 2014b) and to a small team of three robots (Duarte et al., 2014a), and (iii) we show that combining evolved and preprogrammed control allows system designers to automatically synthesize self-organized behaviors and then compose control for large-scale maritime missions based on these behaviors.

Background

Multirobot systems are well suited to tasks that require distributed sensing and/or action. Furthermore, the degree of robustness and reliability of the system is potentially high given the inherent redundancy at the unit-level, and hardware failures often have only limited impact on performance of the system as a whole (Farinelli et al., 2004). Manual design of decentralized control for large-scale multirobot systems has, however, proven difficult, since a set of microscopic behavioral rules that give rise to the desired macroscopic, self-organized behavior cannot be derived in the general case (Dorigo et al., 2004).

ER has been used as an alternative to manual programming of decentralized robotic control, since it allows for the automatic synthesis of control and for the self-organization of behavior. An evolutionary algorithm (Goldberg, 1989) optimizes a controller, which is copied to every robot in the system. The algorithm fine-tunes the microscopic rules guiding the individual robots based on the resulting macroscopic behavior, and thus removes the need for manual specification of low-level control (Floreano and Keller, 2010). Artificial neural networks (ANN) are widely (but not exclusively) used in ER as robotic controllers. ANNs provide evolution with a relatively smooth search space (Nolfi and Floreano, 2000), are able to represent general and adaptive solutions (Floreano and Mondada, 1994), and are able to tolerate noise, which is inherent to many real-world sensors and actuators (Kam-Chuen et al., 1996).

In the domain of multirobot systems, evolutionary techniques have been used in a variety of contexts, from robotic soccer (Uchibe and Asada, 2006), to collective transport (Groß and Dorigo, 2009), and to establish and maintain data links with swarms of flying robots (Hauert et al., 2009).

Notable examples of projects in the domain of swarms of aquatic robots are CoCoRo (Schmickl et al., 2011), which focuses on synthesis of control for underwater robots, and ASSISIbf (Halloy et al., 2013), which aims to develop communication channels between robots and animals (fish and honeybees, in particular). While these projects rely on bioinspired control systems, they are only tangentially related

with evolved ANN-based control, such as the one presented in this paper. Although the use of ER in aquatic robots has not yet been widely explored, a few studies have approached the subject. Some examples include the evolution of locomotion behaviors (Ijspeert et al., 2007; Meyer et al., 2003), station keeping for an underwater robot (Moore et al., 2013), and a neuroaugmenting approach to the evolution of centralized control for a small team of UAVs (Praczyk, 2014).

Over the years, researchers have identified certain challenges associated with the application of ER. One of the most prevalent challenges concerns bootstrapping the evolutionary process in complex tasks. If controllers for a relatively complex task are sought, evolution may be unable to find a fitness gradient that leads to adequate solutions (Nelson et al., 2009). Another challenge is the use of evolved control in real robotic hardware. Except for a few cases in which evolution is conducted onboard real hardware (see, for instance (Watson et al., 1999)), the evolution of robotic controllers is conducted offline, in simulation, due to the large number of evaluation necessary to obtain capable controllers. Simulation-specific features, which are not present in the real world, may be exploited by evolution. As a consequence, the process of transferring evolved controllers to real robotic hardware, known as crossing the reality gap, typically fails to preserve the level of performance achieved in simulation (Jakobi, 1997).

Unless very simple tasks are considered, it is difficult to foresee which evolutionary setup might be suitable for solving a particular task (Christensen and Dorigo, 2006). Between the controller, fitness function and evolutionary algorithm, many different combinations of settings are possible. It then becomes necessary to run the computationally intensive evolutionary process, often multiple times with different initial conditions, to assess if a particular setup produces useful solutions. This leads to a time-consuming trial-anderror process. While a few studies have been conducted in which evolution was applied in a more engineered-oriented manner, such attempts have, so far, been ad-hoc (see Groß et al. (2006) for an example). Techniques such as incremental evolution (Mouret and Doncieux, 2008) and taskdecomposition (Lee, 1999; Whiteson et al., 2005) have been proposed, but such approaches do not address the semiautomatic synthesis of behavior in a systematic way.

Our hybrid approach to the synthesis of behavioral control addresses these issues by applying systematic sub-division of complex tasks into simpler sub-tasks. By evolving behaviors for simpler sub-tasks, the main challenges of ER can be overcome. On the one hand, if a task is too complex for solutions to be evolved, the sub-division still allows us to apply ER techniques to sub-tasks where manually programming solutions is difficult. On the other hand, the sub-division enables the combined use of preprogrammed and evolved control in an engineering-centric approach (Silva et al., 2014), giving more control over the final solutions

to human designers than with traditional evolutionary techniques. Preprogrammed control can also be used when certain robot-environment interactions are too difficult to accurately model in simulation, as in the case of tasks that require fine sensorimotor coordination (Duarte et al., 2014b), which can prevent controllers from successfully crossing the reality gap. Moreover, manually programmed behaviors can easily be scaled to long-term operation, while evolved controllers are not typically evolved for long-duration tasks carried out in large physical spaces, due to the computational cost associated with evolution of controllers in such conditions.

Proposed Solution

In this section, we describe the hardware model and software platform necessary to realize the robotics system proposed in this paper. The conceptual aquatic drones used in our system are relatively small (length: 1 m), inexpensive (< \$1000) and equipped with a number of sensors, such as a camera, GPS, and short-range communication equipment. The aquatic drones are able to move at speeds up to 10 km/h and are equipped with batteries that operate between two and ten hours depending on motor usage. Development and testing of a hardware platform that meets these requirements is being carried out as part of our ongoing work. The local communication system will add the capability to exchange data between drones, such as their GPS location and mission-related information. In this way, communication enables the drones to adjust their behavior based on the actions of other drones, and to coordinate their collective behavior. The capacity to sense nearby robots allows, for instance, drones to maintain coverage of a patrol zone, even when some of the drones experience hardware failure or leave to execute other tasks.

In our approach, behavior primitives are evolved for particular tasks that the drones need to execute during a mission, such as navigation, patrolling, and intruder detection. In this way, a collection of evolved behaviors is built. The individual controllers are then combined hierarchically using behavior arbitrators, which are decision nodes that delegate control to their sub-controllers (Duarte et al., 2014b). The controllers can have multiple hierarchical layers of both evolved and preprogrammed nodes, allowing for detailed control over behavior. By combining evolved behaviors with top-level preprogrammed control, we rely on evolution to find behaviors based on self-organization, while letting the system designer define the conditions that trigger the execution of the different behaviors.

By using such an engineering-centric approach, behaviors can be added or removed based on the mission's characteristics without the need for additional computational-intensive processes: if a particular mission does not require, for instance, intruder detection, that behavior can be easily removed from the top-level preprogrammed control, and the existing manual rules can be modified and tested in various simulated scenarios before real drones are deployed.

Simulation and Experimental Setup

We use JBotEvolver (Duarte et al., 2014c), an open-source, multirobot simulation platform, and neuroevolution framework, for evolution of behavioral control. Each neuralnetwork-based controller in our experiments was evolved using a simple generational evolutionary algorithm with parameter values that have led to the synthesis of good solutions in previous studies (Duarte et al., 2014b). We use a population of 100 genomes, and the fitness score assigned to each genome is the mean score obtained in 10 simulations with different initial conditions. The five highest scoring genomes are copied directly to the next generation. Another 19 copies of each genome are made and mutation is applied to each gene with a probability of 10%. A Gaussian offset with a mean of 0 and a standard deviation of 1 is applied when a gene undergoes mutation. Ten evolutionary runs were conducted for each behavior, and each run lasted 100 generations. Genomes consist of floating-point alleles that encode the parameters of a continuous-time recurrent neural network with one hidden layer of fully-connected neurons (Beer and Gallagher, 1992). All controllers evolved for the experiments presented in this paper have five hidden neurons.

Each drone was equipped with a number of different sensors: (i) four drone sensors with a range of 200 m, (ii) a way-point sensor, which indicates the orientation and distance of a selected GPS waypoint, (iii) four patrol boundary sensors with a range of 50 m, (iv) a Boolean boundary sensor which indicates whether the drone is inside the patrol zone or not, and (v) an intruder sensor, which detects intruders up to a range of 100 m in front of the drone using the camera. All sensors have an opening angle of 90° , and each sensor feeds the ANN with a value in the interval [0,1], depending on the distance of the closest object inside its field-of-view. Since the camera does not collect depth data, it feeds the ANN with (i) an input that indicates whether an intruder is being detected or not, and (ii) an input that indicates the relative direction to the intruder, in case an intruder is detected.

The drone sensors are implemented by combining local communication with GPS position, while the boundary sensors are implemented based on GPS position and a set of GPS coordinates. The four sensors of each type provide drones with omnidirectional drone sensing and omnidirectional boundary sensing up to the sensors' respective ranges. The ANN controller's output layer has three output neurons: two outputs control the speed of both motors, while a third *stop motors* output turns off the motors if its activation is over a threshold of 0.5.

Experiments and Results

To test the proposed approach, we chose a maritime patrolling and intruder detection task where a swarm of aquatic

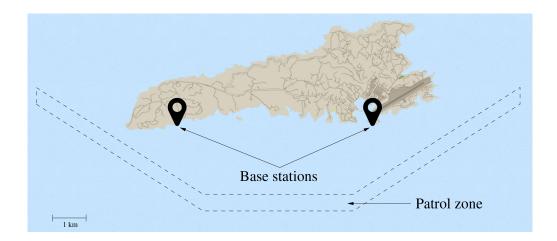


Figure 1: Map of the island of Lampedusa in the Mediterranean Sea, with a 20 km by 0.5 km patrol zone. Drones are deployed from two base stations to a random location inside the patrol zone.

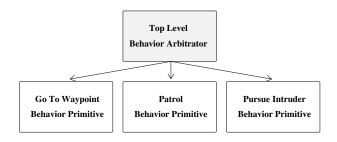


Figure 2: Representation of the hierarchical controller, with one preprogrammed behavior arbitrator and three evolved behavior primitives.

drones must remain within a previously designated patrol zone and pursue intruders that try to cross it. The patrol zone is defined by a polygon, in which the vertices are GPS coordinates. Drones are configured with the mission-specific GPS coordinates prior to deployment. The drones are initially located on one of two base stations, to which they must eventually return in order to recharge their batteries. For the complete mission, we used a patrol zone with a size of 20,000 m by 500 m. Such a patrol zone would allow the coverage of the south coast of the Italian island of Lampedusa (see Figure 1), a major hub for illegal migration from Tunisia and Libya to Italy (Coppens, 2013).

For the complete task, we use three evolved behavior primitives: "Go To Waypoint", "Patrol", and "Pursue Intruder". The behavior primitives were then combined in a mission controller using a top-level preprogrammed behavior arbitrator (see Figure 2). The evolution of the behavior primitives is discussed below.

Evolved behaviors

Go to waypoint: In many maritime missions, it is necessary for drones to move to a specific location, indicated by

a GPS waypoint, without colliding with obstacles or nearby drones. The waypoints can be configured before the drone is deployed, or they can be discovered autonomously during a mission.

Controllers for the "Go To Waypoint" behavior were evolved using a single drone in an environment with an additional 50 static drones that posed as obstacles. The static drones were randomly placed between the starting position and the destination waypoint in a band with a width of 50 m, and the evolving drone had to reach a waypoint within 10,000 control cycles (equivalent to 1,000 seconds). The waypoint was placed from 500 m to 1,000 m from the drone's starting position at a random orientation, with respect to the drone. The evaluation function rewarded controllers for (i) getting close to the waypoint, and (ii) stopping within 3 m of the waypoint. The evaluation function also penalized controllers for colliding with the static drones:

$$f = -10^{-2} \cdot P + \begin{cases} \frac{D-d}{D} & \text{far from waypoint} \\ \\ 2 - \frac{C-s}{C} & \text{near waypoint,} \end{cases}$$

where P is the number of control cycles that the drone collided with static drones, D is the initial distance from the drone to the waypoint, d is the current distance from the drone to the waypoint, C is the total number of control cycles, and S is the number of control cycles where the drone is stopped within 3 m of the waypoint. Seven of the ten evolutionary runs conducted for the "Go To Waypoint" behavior evolved solutions that shut down the motors after reaching the waypoint. In all seven successful solutions, the drone moves at full speed toward the waypoint while evading static drones. After arriving at the waypoint, the drone then shuts down the motors using the S0 motors output of the ANN.

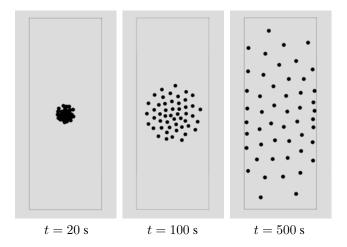


Figure 3: Evolved "Patrol" behavior with 50 drones in a patrol zone with a size of 300 m \times 800 m at different times. In the figure, the size of the drones has been increased relative to the size of the patrol zone for clarity.

Patrol: Once inside the patrol zone, the drones should disperse to cover the zone evenly while remaining inside its pre-defined boundaries. Controllers were evolved for the "Patrol" behavior by placing 50 drones near the center of a rectangular zone with sizes varying from 250 m to 1,500 m from sample to sample. Controllers were evolved for 5,000 control cycles (equivalent to 500 seconds), and each drone had an individual copy of the evaluated ANN controller. The controllers were evaluated at each control cycle based on their distance to other drones and patrol zone boundaries, and on the energy spent:

$$g_i = \sum_{r=1}^{R} \frac{\frac{2}{3} \cdot (1 - max(\alpha_{ir})) + \frac{1}{3} \cdot (1 - p_{ir})}{R},$$

where α_{ir} is the set of activations of drone r's boundary sensors and drone sensors at control cycle i, p_{ir} is the percentage of power applied to the motors, and R is the number of drones. If any of the drones is outside the patrol zone at control cycle i, the value of g_i is set to 0. The fitness obtained in a sample was the mean of g_i during the sample. The evolved controllers move outward in a circle until they stop sensing nearby drones. When the patrol zone is small and drones are able to reach its boundaries before the sample terminates, they stop moving outward and instead optimize the spacing between one another. An example of this behavior can be seen in Figure 3.

Pursue intruder: When an intruder is detected by a drone's camera, the drone should attempt to follow it without crossing the boundaries of the patrol zone. Since the drones' cameras are unable to collect depth data, a minimum of two drones should pursue an intruder in order to triangulate its position. For the evolution of the "Pursue Intruder"

behavior, an intruder attempted to cross a patrol zone. The size of the patrol zone varied in each sample from 250 m to 1,000 m. The 50 drones, each with a copy of the controller being evaluated, started evenly dispersed inside the zone. The intruder moved at a fixed speed of 10 km/h. The controllers were evaluated according to the number of control cycles in which the intruder was detected. Furthermore, g_i was reused in order to reward dispersion of the drones, optimize power usage, and to obtain solutions in which the drones remain inside the patrol zone:

$$h_i = g_i + min(D_i, 2),$$

where D_i is the number of drones detecting an intruder in control cycle i. The fitness obtained in a sample was the mean of h_i during the sample. The evolved controllers pursue the intruder as soon as it is detected, and remain static otherwise. If several drones are pursuing the intruder, some of them cease the pursuit, typically leaving only two. As soon as the intruder exits the patrol zone, the drones stop the pursuit and adjust their positions with respect to neighboring drones.

Preprogrammed arbitrator

After the behavior primitives had been evolved, we combined them using a simple state-based preprogrammed arbitrator (see Figure 4). The arbitrator determines which behavior should be active at any given time, depending on the current state and the drone's sensory inputs. When an intruder is detected, the drone alerts other drones within a 200 m radius. The alerted drones automatically activate the "Pursue Intruder" behavior. If a drone is in the "Pursue Intruder" behavior and has not seen an intruder for the past five minutes, it reverts to the "Patrol" behavior.

The amount of energy necessary to return to the base station is estimated based on the drone's distance to the base

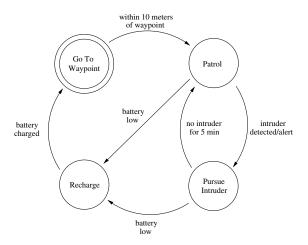


Figure 4: A state machine representing the preprogrammed mission controller.

station. If the battery reaches the estimated limit, the drone returns to the base station in order to recharge its battery.

Scalability tests

We conducted scalability tests with the final controller in a large-scale patrolling and intruder detection task. In the beginning of the experiment, a drone is deployed from each of the two base stations to a random position inside the patrol zone every ten seconds. Each base station deploys drones to one half of the patrol zone. The drones have a battery life of ten hours of stand-by use, or two hours of full throttle. When back at the base station, the drones recharge their batteries for a period of 30 minutes. Two hours into the experiments, intruders begin to cross the patrol zone at random locations every 30 minutes (a total of 44 crossings). The simulation is run for a total of 24 simulated hours.

We ran experiments in ten different scenarios with a number of drones varying from 100 to 1,000 at increments of 100 drones. The simulation was run in a patrol zone with a size of 20,000 m by 500 m, as shown in Figure 1.

The performance observed for the different swarm sizes can be seen in Figure 5. By increasing the number of drones, the number of detected intruders increases until it reaches a maximum of 100% detection rate with 500 drones, which corresponds to a density of 50 drones/km². Another indicator of performance is the amount of time that the intruders are being pursued by either (i) one drone, or (ii) two or more drones. Ideally, an intruder should be pursued by at least two drones in order for its position to be correctly triangulated. Although the detection rate peaked at a swarm size of 500 drones, we can observe a slight improvement in the proportion of time that two or more drones pursue an intruder as we deploy more drones. The mean amount of time until an intruder has been detected after entering the patrol zone decreases as more drones are deployed. With 100 drones the

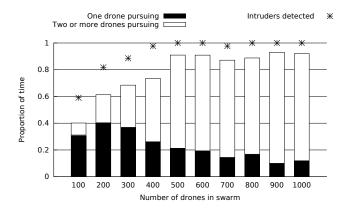


Figure 5: The figure shows both the percentage of detected intruders as points, and the percentage of time that the intruders were pursued by (i) one drone, and (ii) two or more drones, as histograms for the different simulated scenarios.

mean amount of time was 62 s, with 500 drones the mean amount of time was 19 s, and with 1,000 drones the average time was 16 s.

In the scenario with 500 drones, the hybrid controllers successfully completed the task by detecting 44 out of the 44 intruders. In total, intruders were present in the patrol zone for 132 minutes, of which they were detected by a single drone for 28 minutes, and by two or more drones for 92 minutes (21% and 70% of the total time, respectively). In the final ten hours, the system reached an equilibrium in which a mean of 79% of the drones were patrolling, 14% were returning to the base or recharging, 6% were going from the base to the patrol zone, and 1% were pursuing an intruder (see Figure 6). We conducted the same analysis for the setup with 1,000 drones and observed similar results.

Conclusions and Future Work

In this paper, we applied a hybrid approach, based on a combination of evolved and preprogrammed control, to the synthesis of controllers for large swarms of aquatic drones. Controllers for three different behavior primitives were evolved: "Go To Waypoint", "Patrol", and "Pursue Intruder". The controllers were then combined using a state-based preprogrammed behavior arbitrator in order to execute a patrolling and intruder detection task.

The use of ER techniques for the synthesis of self-organized control is particularly relevant for large-scale multirobot systems based on decentralized control, since it is often challenging to manually design behavioral rules for the individual robots. Evolutionary techniques, however, are difficult to apply in complex tasks, provide little control over the solutions evolved, and require a considerable amount of time to evolve controllers for large robotic swarms. By dividing the task into simpler sub-tasks, it is possible to evolve robust and scalable behaviors that solve different parts of the

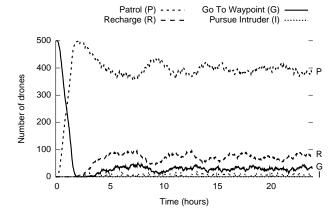


Figure 6: Plot of the states of the drones' preprogrammed behavior arbitrators in the scenario with 500 deployed drones over a period of 24 hours of simulation.

task. The controllers evolved for the sub-tasks can then be combined using a preprogrammed behavior arbitrator that controls when each behavior primitive should be executed. Furthermore, evolved behavior primitives can potentially be reused in various types of missions.

In our ongoing work, we are exploring ways in which inter-drone communication can be further exploited in maritime scenarios. Moreover, we are preparing our hardware testbed for proof-of-concept missions based on the approach proposed in this paper. Our long-term goal is to have a software and hardware platform for swarm-based maritime missions.

Acknowledgements

This work was partly supported by FCT – Foundation of Science and Technology under grants SFRH/BD/76438/2011, PEst-OE/EEI/LA0008/2013 and EXPL/EEI-AUT/0329/2013.

References

- Bayindir, L. and Şahin, E. (2007). A review of studies in swarm robotics. *Turkish Journal of Electrical Engineering & Computer Sciences*, 15(2):115–147.
- Beer, R. D. and Gallagher, J. C. (1992). Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, 1(1):91–122.
- Carling, J. (2007). Migration control and migrant fatalities at the Spanish-African borders. *International Migration Review*, 41(2):316–343.
- Christensen, A. L. and Dorigo, M. (2006). Evolving an integrated phototaxis and hole-avoidance behavior for a swarm-bot. In *Proceedings of the 10th International Conference on the Simulation & Synthesis of Living*

- Systems (ALIFE), pages 248–254. MIT Press, Cambridge, MA.
- Coppens, J. (2013). The Lampedusa disaster: How to prevent further loss of life at sea? *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation*, 7(4):589–598.
- Dorigo, M., Trianni, V., Şahin, E., Groß, R., Labella, T., Baldassarre, G., Nolfi, S., Deneubourg, J., Mondada, F., Floreano, D., and Gambardella, L. M. (2004). Evolving self-organizing behaviors for a swarm-bot. *Autonomous Robots*, 17(2):223–245.
- Duarte, M., Oliveira, S. M., and Christensen, A. L. (2014a). Evolution of hierarchical controllers for multirobot systems. In *Proceedings of the 14th International Conference on the Synthesis & Simulation of Living Systems (ALIFE)*. MIT Press, Cambridge, MA. In press.
- Duarte, M., Oliveira, S. M., and Christensen, A. L. (2014b). Evolution of hybrid robotic controllers for complex tasks. *Journal of Intelligent and Robotic Systems*. In press.
- Duarte, M., Silva, F., Rodrigues, T., Oliveira, S. M., and Christensen, A. L. (2014c). JBotEvolver: A versatile simulation platform for evolutionary robotics. In Proceedings of the 14th International Conference on the Synthesis & Simulation of Living Systems (ALIFE). MIT Press, Cambridge, MA. In press.
- Farinelli, A., Iocchi, L., and Nardi, D. (2004). Multirobot systems: a classification focused on coordination. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(5):2015–2028.
- Floreano, D. and Keller, L. (2010). Evolution of adaptive behaviour in robots by means of Darwinian selection. *PLoS Biology*, 8(1):1–8.
- Floreano, D. and Mondada, F. (1994). Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot. In *From Animals to Animats 3: Proceedings of the 3rd International Conference on Simulation of Adaptive Behavior (SAB)*, pages 421–430, MIT Press, Cambridge, MA.
- Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Boston, MA.
- Groß, R., Bonani, M., Mondada, F., and Dorigo, M. (2006). Autonomous self-assembly in swarm-bots. *IEEE Transactions on Robotics*, 22(6):1115–1130.
- Groß, R. and Dorigo, M. (2009). Towards group transport by swarms of robots. *International Journal of Bio-Inspired Computing*, 1(1–2):1–13.

- Halloy, J., Mondada, F., Kernbach, S., and Schmickl, T. (2013). Towards bio-hybrid systems made of social animals and robots. In *Proceedings of the 2nd Inter*national Conference on Biomimetic and Biohybrid Systems (LM), pages 384–386. Springer, Berlin, Germany.
- Hauert, S., Zufferey, J.-C., and Floreano, D. (2009). Evolved swarming without positioning information: an application in aerial communication relay. *Autonomous Robots*, 26(1):21–32.
- Ijspeert, A. J., Crespi, A., Ryczko, D., and Cabelguen, J.-M. (2007). From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315(5817):1416–1420.
- Jakobi, N. (1997). Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adaptive Behavior*, 6(2):325–368.
- Kam-Chuen, J., Giles, C., and Horne, B. (1996). An analysis of noise in recurrent neural networks: convergence and generalization. *IEEE Transactions on Neural Networks*, 7:1424–1438.
- Lee, W.-P. (1999). Evolving complex robot behaviors. *Information Sciences*, 121(1-2):1–25.
- Lutterbeck, D. (2004). Between police and military the new security agenda and the rise of gendarmeries. *Cooperation and Conflict*, 39(1):45–68.
- Lutterbeck, D. (2006). Policing migration in the mediterranean. *Mediterranean Politics*, 11(1):59–82.
- Manley, J. E. (2008). Unmanned surface vehicles, 15 years of development. In *Proceedings of OCEANS 2008*, pages 1–4. IEEE Press, Piscataway, NJ.
- Meyer, J.-A., Doncieux, S., Filliat, D., and Guillot, A. (2003). Evolutionary approaches to neural control of rolling, walking, swimming and flying animats or robots. In Duro, R., Santos, J., and Graña, M., editors, *Biologically Inspired Robot Behavior Engineering*, pages 1–43. Springer, Berlin, Germany.
- Monzini, P. (2007). Sea-border crossings: The organization of irregular migration to italy. *Mediterranean Politics*, 12(2):163–184.
- Moore, J. M., Clark, A. J., and McKinley, P. K. (2013). Evolution of station keeping as a response to flows in an aquatic robot. In *Proceeding of the 15th An*nual Conference on Genetic and Evolutionary Computation Conference (GECCO), pages 239–246. ACM Press, New York, NY.

- Mouret, J.-B. and Doncieux, S. (2008). Incremental evolution of animats' behaviors as a multi-objective optimization. In *Proceedings of the 10th International Conference on the Simulation of Adaptive Behaviour (SAB)*, pages 210–219. Springer, Berlin, Germany.
- Nelson, A. L., Barlow, G. J., and Doitsidis, L. (2009). Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57(4):345–370.
- Nolfi, S. and Floreano, D. (2000). *Evolutionary robotics:* The biology, intelligence, and technology of self-organizing machines. MIT Press, Cambridge, MA.
- Praczyk, T. (2014). Using augmenting modular neural networks to evolve neuro-controllers for a team of underwater vehicles. *Soft Computing*, pages 1–16.
- Schmickl, T., Thenius, R., Moslinger, C., Timmis, J., Tyrrell, A., Read, M., Hilder, J., Halloy, J., Campo, A., Stefanini, C., et al. (2011). CoCoRo–The Self-Aware Underwater Swarm. In *Proceedings of the 5th IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASO)*, pages 120–126. IEEE Press, Piscataway, NJ.
- Schwing, R. P. (2007). Unmanned aerial vehiclesrevolutionary tools in war and peace. Technical report, U.S. Army War College, Carlisle Barracks, Carlisle, PA.
- Silva, F., Duarte, M., Oliveira, S. M., Correia, L., and Christensen, A. L. (2014). The case for engineering the evolution of robot controllers. In *Proceedings of the 14th International Conference on the Synthesis & Simulation of Living Systems (ALIFE)*. MIT Press, Cambridge, MA. In press.
- Uchibe, E. and Asada, M. (2006). Incremental coevolution with competitive and cooperative tasks in a multirobot environment. *Proceedings of the IEEE*, 94(7):1412–1424.
- Watson, R., Ficici, S., and Pollack, J. (1999). Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In *Proceedings of the Congress on Evolutionary Computation (CEC)*, pages 335–342. IEEE Press, Piscataway, NJ.
- Whiteson, S., Kohl, N., Miikkulainen, R., and Stone, P. (2005). Evolving keepaway soccer players through task decomposition. *Machine Learning*, 59(1):5–30.
- Yan, R.-j., Pang, S., Sun, H.-b., and Pang, Y.-j. (2010). Development and missions of unmanned surface vehicle. *Journal of Marine Science and Application*, 9(4):451–457.

Identifying Necessary Conditions for Open-Ended Evolution through the Artificial Life World of Chromaria

L. B. Soros and Kenneth O. Stanley

Department of EECS (Computer Science Division) University of Central Florida, Orlando, FL 32816 lsoros@cs.ucf.edu, kstanley@cs.ucf.edu

Abstract

A full understanding of open-ended evolutionary dynamics remains elusive. While artificial life worlds have been proposed to study such dynamics and tests have been devised to try to detect them, no theory yet has enumerated the key conditions that are essential to inducing them. The aim of this paper is to further such an understanding by hypothesizing four conditions that are essential for open-ended evolution to prosper. Of course, any such conditions must be satisfied by nature (the clearest example of an open-ended domain), but we do not know the scope or range of possible worlds that could achieve similarly impressive results. To complement the hypothesized conditions, a new artificial life world called Chromaria is introduced that is designed explicitly for testing them. Chromaria, which is intended to deviate from Earth in key respects that highlight the breadth of possible worlds that can satisfy the four conditions, is shown in this paper to stagnate when one of the four conditions is not met. This initial controlled experiment thereby sets the stage for a broad research program and conversation on investigating and controlling for the key conditions for open-ended evolution.

Introduction

The particular properties that characterize open-ended evolution are tricky to pin down and often lack consensus (Bedau et al., 1998; Channon, 2003, 2006; Juric, 1994; Maley, 1999). For example, it has been variously characterized as the continual production of either novel (Lehman and Stanley, 2011; Standish, 2003) or adaptive (Bedau et al., 1998) forms. Yet despite the difficulty of precisely pinpointing this phenomenon, a major goal of artificial life (alife) research remains to observe open-ended evolution in an alife simulation (Bedau et al., 2000). In fact, there is little doubt that no algorithm yet devised has fully reproduced it. Even with milestone artificial worlds like Geb (Channon, 2003, 2006) that have passed tests designed to detect particular signatures of open-endedness (Bedau et al., 1998), no scientist has suggested that any system today reproduces the full generativity of nature in all its glory, which raises a fascinating question: why not? What aside from eons of time (which likely is not the sole ingredient missing from artificial worlds so far) could ignite the fire of an open-ended complexity explosion?

The aims of this paper are to provoke progress towards understanding why it has not happened so far by (1) proposing a set of four conditions that are hypothesized to be essential for triggering a genuinely open-ended evolutionary process, and (2) showing how these conditions can be tested using a novel artificial life world called *Chromaria*, which implements all four conditions. An important theme of the proposed conditions is that they are not only satisfied by Earth, but in principle would be satisfied by any abstract artificial system that is to achieve open-ended dynamics (including systems radically different from Earth). Thus evaluating these four conditions entails a broad research program that a single conference paper cannot comprehensively address. Nevertheless, as an initial step in this direction, this paper presents an experiment wherein one of the conditions is controlled in Chromaria to demonstrate a general methodology for testing the conditions for open-endedness, and also to show that even one condition's absence can profoundly and observably incapacitate evolution. The hope in the longer term is eventually to examine all four conditions, including in alife worlds besides Chromaria, and moreover to initiate a reinvigorated discussion on the essential conditions for evolution of the scale observed on Earth.

Background: Open-Ended Evolution

Attempts to achieve open-ended evolution in alife often center on artificial worlds or simulators inspired by some aspect of natural evolution. Among the first is Tierra (Ray, 1992), which consists of a virtual machine that executes machine code. Tierran creatures are programs (i.e. sets of machine code instructions) that are stored in RAM and compete for CPU cycles. The ideas in Tierra later inspired Avida (Ofria and Wilke, 2004), where unlike in Tierra CPU time is allocated proportionately to a fitness measure called *merit* that is based on a creature's ability to perform various computations. Avida has shown in a landmark study that the evolution of complex behaviors can stem from the evolution of simpler ones (Lenski et al., 2003).

The work of Sims (1994) on evolving three-dimensional creature morphologies has inspired its own less abstract

genre of alife world. For example, the Division Blocks (Spector et al., 2007) environment consists of square islands on a square ocean, with a sun that circles the world and energizes evolving three-dimensional creatures that sometimes evolve altruistic behaviors. In another such three-dimensional world, Evosphere (Miconi and Channon, 2005), creatures coevolve nontrivial strategies through direct physical combat.

Another popular genre includes PolyWorld (Yaeger, 1994), where neural network-based organisms that were shown to increase in complexity forage and fight on a twodimensional plane. Geb (Channon, 2003, 2006) is a similar (yet independently conceived) world consisting of a toroidal, two-dimensional grid. Additionally, Geb is the first and only alife system thus far to qualify as unbounded according to the activity statistics classification system (Bedau et al., 1998; Channon, 2003, 2006). This system measures the persistence of advantageous genotypes over evolution, following from the assumption that a gene that is not eliminated by natural selection is in effect a beneficial adaptation. However, there is debate as to whether or not this assumption is valid (Juric, 1994; Miconi, 2008). Furthermore, it is not clear that achieving unbounded adaptation is equivalent to achieving a complexity explosion. It remains unsettled the extent to which activity statistics capture the essence of open-ended phenomena observed on Earth.

Bearing in mind the controversy surrounding the definition of open-endedness, for simplicity this paper follows the definition proposed by Standish (2003), which is that *open-endedness depends fundamentally on the continual production of novelty*. The continual production of novelty furthermore *entails* increasing complexity because all the possibilities that exist at any given level of complexity will eventually be exhausted in a never-ending process. Therefore, experiments in this paper quantify open-endedness based on the ongoing generation of novel behaviors.

While a variety of alife worlds have been implemented and studied, critics describe them as "lacking" in methodology and theory (Miconi, 2008). Among the limited work aiming for such a theory, Conrad and Pattee (1970) make an initial attempt to isolate minimal conditions for evolution in ecosystems and empirically test them via simulation. Holland (1994) similarly investigates the necessary conditions for emergent phenomena in complex adaptive systems. In contrast, the present work aims to provide concrete, testable conditions for *open-ended* evolution in particular. Furthermore, the hope is to understand how such a phenomenon can be provoked in all possible evolutionary domains, of which nature is only one.

Hypothesized Necessary Conditions

The aim in this paper is to initiate a new direction in identifying the set of necessary conditions that a domain must satisfy to support open-ended evolution. This goal is ambitious

because while evolution in nature must satisfy such conditions, no artificial domain yet devised exhibits the same kind of astronomical complexity explosion. Thus it follows that if any set of proposed conditions is credible, natural evolution must satisfy all of them while every artificial system so far devised likely falls short of meeting at least one condition. Furthermore, unless evolution in nature is the only openended system that is theoretically possible (which would be a disappointing conclusion for alife), such a set of conditions should admit conceivable worlds far different from our own, thereby elucidating what from nature is genuinely essential to provoking such a process, and what simply lends character to nature. Such differences might then point the way to open-ended domains different from those typical in alife (such as the one later proposed in this paper) while also explaining why many domains that seem natural nevertheless can be predicted to fall short.

It is important to note up front that a domain can only support *open-ended* evolution if it is generally suitable to evolution in general. That is, any artificial or natural evolutionary system must satisfy certain minimal prerequisites to have *any* success at all (open-ended or not). These prerequisites include a good genetic representation (tightly coupled with the phenotype space) (De Jong, 2006), a sufficiently large world for every individual to be evaluated, and some initial *seed* (like the first cell on earth) or starting point (such as a random initial population) from which evolution begins. Assuming that these general prerequisites are met, the main hypothesis is that four necessary conditions for open-ended evolution are as follows:

Condition 1: A rule should be enforced that individuals must meet some minimal criterion (MC) before they can reproduce, and that criterion must be nontrivial. The role of the MC is to ensure that a minimal level of complexity must always be maintained by every viable organism, thereby ensuring that the population can never degenerate into trivial behaviors. On Earth, for example, individuals become eligible to reproduce only by developing and maintaining functional reproductive apparatuses. No lineage can persist that does not maintain this nontrivial capability, which is the MC on Earth. In worlds that are unlike Earth, however, fundamentally different MC are conceivable; for example, the criterion need not concern the mechanics of reproduction at all. That is, offspring could be created by the system, as is common in evolutionary algorithms (EAs), without any reproductive apparatus in the individuals themselves. There simply must be some meaningful limit on which individuals can reproduce, thereby ensuring that the results will remain interesting. For example, every individual could be required to perform a particular complex task. To be meaningful or nontrivial, the criterion should involve interacting with the world in some way. If the MC is too trivial, the results of evolution will be uninteresting. On the other hand, if it is too demanding then the search will be too restricted. Interestingly, in evolutionary computation, with a few exceptions (Lehman and Stanley, 2010), most EAs implement no such MC: usually all individuals have at least some small probability of reproduction (De Jong, 2006).

This condition implies a necessary **corollary**: The initial seed (from which evolution begins) must itself meet the MC and thereby be nontrivial enough to satisfy Condition 1. Otherwise, if evolution began without any individuals who satisfy the MC, then by Condition 1 no one would be allowed to reproduce and the experiment would end. This corollary further diverges from traditional EAs (and even alife worlds), which often begin with a random population. If no individual in the initial population of such an experiment meets a nontrivial MC then the algorithm could not satisfy Condition 1. In fact, this corollary shows that obtaining the starting seed (such as the first cell on Earth, which already began with a reproductive apparatus) is a challenge in its own right that must be confronted for open-ended evolution even to initiate.

Condition 2: The evolution of new individuals should create novel opportunities for satisfying the MC. This condition is important because it ensures that there is some way for complexity to increase indefinitely beyond the level of the (relatively simple) initial seed. If evolution is to achieve open-endedness, then it must continually find paths from simpler phenotypes to more complex ones. However, such novel paths will be explored only as long as each link in the chain continues to satisfy the MC. Thus it is critical that new opportunities to satisfy the MC through previously unsupportable strategies continually open up so that evolution can explore paths that lead arbitrarily beyond the initial seed. Furthermore, evolution itself is the only viable generator of such novel opportunities because a human designer could not realistically conceive a ladder of tasks sufficiently rich to continue without limit. The trajectory of biological evolution exhibits many such transitions where new life forms paved the way for further genetic innovation in other lineages. Giraffes, for instance, could not have evolved on Earth before there were trees. In this way, the evolution of trees created an opportunity for evolution later to explore a previously unsupportable path (namely, the path to giraffes) by generating a novel opportunity to satisfy the MC. If the nature of individual interactions is too restrictive (such as predators simply bumping into prey in alife worlds), then this condition may not be possible to satisfy.

Condition 3: Decisions about how and where individuals interact with the world should be made by the individuals themselves. Such decisions determine whether an individual will successfully seek out and exploit novel opportunities for satisfying the MC. Though the MC primarily serves to maintain some degree of complexity, it also creates a coupling between successful phenotypes and the environment. (This coupling follows from the requirement that

a nontrivial MC must involve interacting with the world in some way, such as gathering food for sustenance of the reproductive apparatus in natural evolution.) If an individual cannot choose both its actions and their targets (geographical or otherwise), then the environmental coupling is disrupted and phenotypic evaluation becomes arbitrary. That is, if the individual does not play at least some role in deciding where it interacts, then some kind of oracle would need to determine for the individual its best opportunity to satisfy the MC. However, this oracle would require intimate knowledge of the search space to anticipate all possible future opportunities, most of which could not even exist when the search began. In this sense, any decision made by such an oracle would be effectively arbitrary. Thus no human designer can realistically construct such an oracle. For these reasons, behavioral decisions must be made by the individuals themselves, who thereby decide for themselves which opportunities to exploit (like giraffes heading for the trees).

Condition 4: The potential size and complexity of the individuals' phenotypes should be (in principle) unbounded. In practice, the growth of the phenotype must have some limit (e.g., the size of the universe in the real world); achieving infinite growth would of course require infinite time and infinite space. However, at a practical level, the salient point is that the complexity of the phenotype should not be limited by its representation, as there needs to be room for complexity to increase for the kind of complexity explosion desired in open-ended evolution to be realized.

While the hypothesis is that these four conditions are all necessary for open-ended evolution, whether they are sufficient is left open. Perhaps more conditions will be identified. However, the hope is that the set of necessary and sufficient conditions can be kept as small as possible. That way, the conditions can help to illuminate the elusive fundamental and essential ingredients behind open-ended complexity explosions. To this end, what is omitted can be just as illuminating as what is included. For example, one prominent omission from the four proposed conditions is any reference to traditional fitness. While the fitness function is ubiquitous in much of evolutionary computation (De Jong, 2006), a major problem for open-ended evolution is that what is genuinely novel is significantly harder to formalize than what is better. In effect, we do not know a priori which discoveries lead later to more novel discoveries. Any explicit conception of fitness, wherein some individuals are judged less meritorious relative to others, risks blocking potentially promising paths through the search space. By introducing a mechanism that evaluates individuals without judging them against each other (i.e. the MC), evolution can maintain nontriviality without falling prey to such deception. The benefits of search not driven by explicit objectives has been explored previously in e.g. Lehman and Stanley (2011). Of course, fitness is indeed an important concept in natural evolution (Orr, 2009), but it can be viewed as an *emergent* byproduct of the MC in nature (i.e. to construct a copy of oneself) that changes over time rather than an explicit a priori constraint imposed from outside. By thereby reducing the conditions to rely solely upon the constraint of the MC, the simplest possible hypothesis on the origin of open-ended dynamics can be explored, and fewer assumptions must be satisfied to construct such systems.

At the same time, the conditions can help us to predict which systems can be expected to yield genuine openendedness, and which cannot. For example, any system (1) without a nontrivial MC or without an initial seed satisfying such a MC, (2) in which the means of satisfying the MC are fixed from the start, (3) in which individuals do not choose for themselves with whom or where to interact, or (4) without the ability to increase the size of the genetic representation is by hypothesis *not capable of open-ended dynamics*.

More generally, the interesting potential of such a set of conditions, if right, is that they can admit worlds radically different from nature yet able to exhibit similar open-ended dynamics. By exploring such alternative worlds, as in the experiment described next, ideally we can begin to learn what may be possible someday to achieve by harnessing open-ended dynamics for our own purposes.

Chromaria Experiment

The aim of this experiment is to observe initial hints of openended evolutionary dynamics in an artificial domain that satisfies the four hypothesized necessary conditions, and to show that when one of the conditions is not satisfied, the observed phenomenon is stunted. That way, the main contribution is to suggest a concrete path towards investigating the hypothesized conditions. In addition, by also introducing a world that is intentionally unlike Earth, the implication is that the four conditions admit many possible realizations, of which biology is only one.

The world introduced in this paper, called *Chromaria*, is visually two-dimensional and composed of discrete RGB pixels (Figure 1). The colorful creatures (called *Chromarians*) evolved in this world actively explore it to search for a place to *plant*. Each Chromarian is allowed one planting attempt. If the Chromarian's RGB sensor field (which can sense prior successful planters and the background) satisfies a specific *planting function* involving matching its color (detailed later), then the planting attempt succeeds and the successful creature is eventually allowed to reproduce. Thus the MC in Chromaria, unlike Earth's MC, is to navigate to a position in the world with colors matching the Chromarian's own coloring. If this MC is not met, then the Chromarian is removed without planting and does not reproduce.

Each Chromarian's morphology consists of a twodimensional image composed of RGB pixels. The genetic encoding of this morphology is a compositional pattern producing network (CPPN; Stanley 2007), a neural-network-

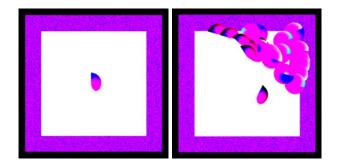


Figure 1: **Chromaria.** Each creature is born at the center of the world (left) and then must find an appropriate place to plant. The color-rich borders initially provide the only viable options, but more emerge as Chromarians continue to plant in the environment (right).

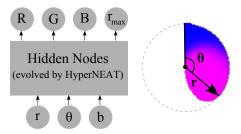


Figure 2: **Morphology-encoding CPPN.** The CPPN encodes both the outline and fill of the Chromarian's morphology. Input b is a bias set to the constant value 1.0.

like representation that generates patterns with regularities such as symmetry, repetition, and repetition with variation. The CPPN used to encode Chromarian morphologies (Figure 2), which is similar to the encoding in Risi et al. (2012), takes polar coordinates r and θ as input. Such polar coordinates define an unambiguous solid border for the body, which would be harder to determine if the inputs were Cartesian. Upon activation, the CPPN returns an r_{max} for each value of θ , which determines the perimeter of the Chromarian's body at that angle. Then every pixel on the interior of this border is queried by the CPPN for the corresponding RGB values at the queried (r, θ) , where r is scaled from [0,49] to [0,1], and θ from [$-\Pi$, Π] to [-1,1]. In this way the CPPN determines both the *shape* (via the r_{max} output) and internal color (via the RGB outputs) of the Chromarian. These characteristics ultimately determine where the Chromarian can successfully plant. By evolving new colors, Chromarians in effect create novel opportunities for new kinds of planters, thereby satisfying Condition 2.

Each Chromarian is equipped with a 10×10 rectangular sensor field that perceives the RGB values (each scaled from the range [0,255] to [-1,1]) of the underlying pixels. This field is centered at the forefront of the Chromarian's body, with half of the pixels falling underneath the body and

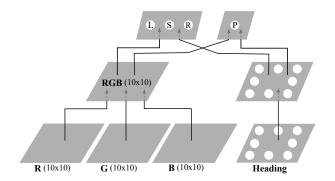


Figure 3: **Behavioral controller.** Each plane represents an array of sensors or neurons. Arrows between planes in this schematic denote *sets* of connections between one plane and another. The input layer contains three individual color fields, which feed into an intermediate integration layer. There is an additional set of heading inputs, which remain uncombined with the color data until the output layer is reached. The four output nodes control behavior. The maximum number of connections in this network (evolved by HyperNEAT) is 30,448.

the rest extending in front of the creature. The exact resolution of the field depends on the creature's morphology; as its length and width increase, the distance between neighboring sensors grows. Note that Chromarians can overlap if they have planted in the same location, in which case the pixels of the most recent planter are sensed. Additionally, each Chromarian is equipped with a heading-sensitive *compass* consisting of 8 pie slice sensors. All sensors are input to a multimodal neural controller (Figure 3), whose weights are encoded using a second CPPN following the HyperNEAT approach to encoding large-scale ANNs with CPPNs (Pugh and Stanley, 2013; Stanley et al., 2009). The output layer, which receives connections from the hidden layers, has four effector nodes corresponding to the Chromarian's requested rotation (L and R), speed (S), and desire to plant itself (P). If the planting node exceeds a threshold, then the Chromarian is immobilized and it never moves again. Otherwise, the rotation and speed nodes determine the Chromarian's next movement. Note that it is through this ability of the Chromarian to decide for itself when and where to plant (based on its senses) that it satisfies Condition 3. HyperNEAT's ability to evolve multimodal neural networks with tens of thousands of connections is what enables creating an artificial life world like Chromaria, where autonomous control decisions are made based on rich full-color sensory input.

Evolution in Chromaria proceeds in two stages. First, there is a preliminary search for an initial seed that satisfies the MC (i.e. an individual that successfully plants itself), followed by the open-ended phase (which proceeds without a particular desired behavior or morphology).

Preliminary search. Satisfying the corollary to Condition 1 (i.e. that evolution must start with an initial seed that satisfies the MC) presents a puzzle: how can the initial seed be obtained? The proposed solution in Chromaria is to begin with a *preliminary search* for it. The search for an initial seed is important because it decides the starting point for subsequent open-ended evolution (and thereby influences which potential paths to complexity evolution will explore). In this experiment, novelty search (Lehman and Stanley, 2011) is the approach chosen to find a successful controller for the initial morphology (starting morphologies were interactively evolved by the authors). Planters discovered by novelty search for particular morphologies are then the seeds in the experiment that initiate open-ended evolution.

Open-ended evolution. Chromarians are evolved using HyperNEAT (Stanley et al., 2009), a neuroevolution method that gradually complexifies its underlying genetic representation over time. Through the indirect CPPN encoding, HyperNEAT can efficiently evolve complex connectivity patterns that reflect the geometry of their inputs. The large HyperNEAT substrate neural networks (up to 30,448 connections) in this experiment provide sufficient space for complexity to increase significantly over the course of a run, thereby satisfying Condition 4. Recall also that each Chromarian contains two CPPNs: one to encode its neural network, and the other to encode its morphology. This experiment uses a modified version of the HyperSharp-NEAT 2.1 implementation of HyperNEAT, which is based on Colin Green's SharpNEAT (Green, 2006). Parameter settings are included with the released code, available at http://eplex.cs.ucf.edu/chromaria/home.

In the unconventional main loop in Chromaria, the Chromarians that have successfully planted most recently are kept in a *parent queue* with maximum size 100. A *current parent* pointer always points to one position in the list. When the simulation begins, the list only contains one Chromarian (i.e. the initial seed found in the preliminary search). The newborn then attempts to plant. Each tick of the simulation (capped at 200 ticks per Chromarian) proceeds as follows:

- 1. The Chromarian's sensors are updated and its controller is activated.
- 2. If the planting effector node is negative, the Chromarian moves according to its other effector nodes. Otherwise, the Chromarian attempts to plant at its current location. This attempt succeeds if the the RGB ratios in the Chromarian's morphology are collectively no greater than 12.5% different from the RGB ratios in its sensor array contents (Figure 4). Furthermore, to ensure that Chromarians must learn to move (to keep the MC nontrivial), they are eliminated if they attempt to plant within a small radius of their starting position.
- 3. If the planting attempt succeeds, the Chromarian generates an offspring. Note that reproductive dynamics in

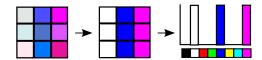


Figure 4: **RGB ratio calculation.** Each pixel of both the morphology and sensor field is placed into one of eight bins: black, white, red, green, blue, yellow, cyan, or magenta. Here, a simple morphology is shown to the left of its binned equivalent and a histogram of the bins. The bins are defined by halving the ranges [0,255] that the R, G, and B component values can take. For instance, any pixel with $R \in [0, \frac{255}{2}]$ (more non-red than red), $G \in [0, \frac{255}{2}]$ (more non-green than green), and $B \in [0, \frac{255}{2}]$ (more non-blue than blue) falls into the black bin because black has values R,G,B = 0,0,0. Once every pixel is binned in this way, color ratios are calculated for each bin by dividing the bin size by the total number of pixels. Ratios are recorded for both the morphology and sensor field. The differences between these ratios for each color are summed to get a *matching value*. If this value is less than 1 (out of 8), the planting function is satisfied.

Chromaria are unlike those in many other alife worlds. The next Chromarian to reproduce is always next in the parent queue. If the current parent is at the end of the queue, the pointer simply wraps back to the start of the queue. In this way, all Chromarians who successfully plant eventually get to reproduce. That is, explicit competition, which is usually central to alife worlds, is intentionally absent from Chromaria (because it still satisfies the four conditions anyway). As soon as an offspring is created from the current parent, it attempts to plant, starting as always from the center of the world. If it succeeds, then it is inserted into the parent queue directly preceding its own parent. Then the next Chromarian in the queue reproduces, and so on. This mechanism of always inserting offspring preceding their parent forces the system to allow every preexisting member of the population to reproduce before a newcomer.

4. Whenever a new Chromarian succeeds at planting, the oldest preexisting member is removed from the population list if the list contains at least 100 members. However, *all* bodies of previously successful planters remain in the world for the duration of evolution.

Chromaria thus satisfies the four hypothesized necessary conditions for open-ended evolution: (1) individuals must satisfy a nontrivial MC (finding a valid location in which to plant) before they can reproduce; (2) individuals can plant within each other, wherein the possible color matchups are unlimited, thereby creating novel opportunities for satisfying the MC; (3) individuals decide themselves where and when to plant based on information from their sensors; and (4) the CPPN encodings for the creatures' morphologies and

controllers have no complexity ceiling; thus complexity has room to increase significantly.

It is also instructive to consider how Chromaria is unlike Earth, which is the canonical example of an open-ended domain. First, the MC in Chromaria decouples the function of reproduction (producing offspring) from the process that allows one to reproduce. Second, there are no predator-prey relationships between any individuals. This absence highlights the variety of conceivable ways that individuals in a non-Earth-based domain can satisfy Condition 3 by creating opportunities for each other. Furthermore, unlike in many alife worlds, there is no explicit competition: Anyone who satisfies the MC will eventually reproduce. This lack of explicit competition, and the implication as suggested by Juric (1994); Lehman and Stanley (2011); Standish (2003) that open-endedness is still possible without it, is the reason that activity statistics (Bedau et al., 1998), which track adaptive evolution, are not the chosen measure in this paper.

Recall that one of the four conditions will be tested in this paper through a controlled experiment in Chromaria to show how Chromaria can serve as a testbed for such investigations. For that purpose, in this initial experiment the second condition is controlled by preventing Chromarians from sensing each other. That way, new opportunities to plant can never arise beyond the preexisting colored border region and white background present at the start of the run (recall that the right to plant is based on the contents of the Chromarians' sensors), violating Condition 2. Five runs each of the **control** and **standard** (i.e. satisfying all four conditions) versions of Chromaria were performed, starting from the same initial seed (discovered by novelty search to plant in a magenta and blue border region) and ending after 50,000 reproductions. Additionally, a second set of ten runs (five control and five standard) was initiated from a different seed evolved for a world with a cyan and blue border region.

Results

The goal of the experiment is to observe the continual production of novel and successful planters in Chromaria, and to show that such continued innovation (hinting at openended evolution) is precluded when Condition 2 goes unmet. The most salient evidence of such a dichotomy between standard and control runs is from observing the worlds themselves. A key benefit of the visual design of Chromaria is to make observing the implications of different variants easy. For this purpose, representative snapshots of Chromaria at equivalent stages in different variants are shown in Figure 5. These snapshots illustrate a clear expansion and diversification (both in terms of planting locations and colors) of the population, which was observed in every standard run but absent from the control runs. Furthermore, distinct color-behavioral groups can be seen gradually emerging and unfolding over the 50,000 reproduction attempts of each world. Importantly, appreciating the full

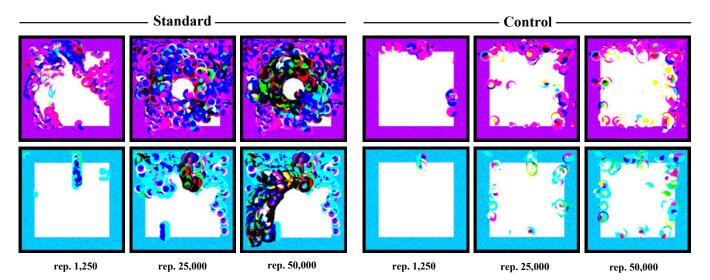


Figure 5: **Representative world snapshots.** Control and standard versions of Chromaria were run through 50,000 reproductions. Here, representative worlds (magenta/blue world at top and cyan/blue at bottom) are shown at various reproduction numbers. While each run followed a different trajectory, *every* standard run exhibited principled growth beyond the initial world state, while *no* control runs did. Thus interactions between individuals are clearly required for open-ended evolution.

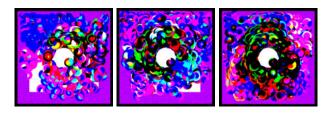


Figure 6: **Typical standard run end states.** At 50,000 reproductions, standard runs in the magenta world typically exhibit circular patterns of growth. However, while such patterns are discernible, differences in the individual snapshots indicate unique trends in individual runs.

breadth of Chromaria requires observing Chromarians in action (i.e. exploring and attempting to plant) because much of the complexity of evolved Chromarians is in their dynamic *behavior*, which is based on their rich sensory inputs. For that purpose highlight videos are available at http://eplex.cs.ucf.edu/chromaria/home.

Another intriguing result is that even though the world is stochastic, the end states among the standard runs exhibit some consistent dynamics (an outcome that could not necessarily be predicted from the start). As shown in Figure 6, the magenta worlds results consistently in ringlike configurations. Mostly-black Chromarians, who are not present at the start, tend to encircle the middle of the world.

To quantify the dramatic difference Condition 2 makes, the diversity of behaviors generated in different variants was measured. The path of an individual Chromarian is represented as a vector of (x, y) coordinates with range [(0,0), (1000, 1000)]. For the purpose of sampling the vari-

ance of behaviors, position is sampled ten times (at every 20 simulation ticks), giving a vector of length 20 for each individual. The breadth of behavioral trajectories in a run can then be characterized by calculating the average variance in position (\overline{var}) at each sampled tick. It is important to note that these behaviors (represented by the vector of all sampled positions) capture both a period of intelligent seeking and then planting once a suitable color destination is identified. Given that successful planting is a nontrivial behavior, this metric captures not just the amount of diversity produced by evolution, but the amount of interesting or non*trivial* diversity. In the blue world, the average \overline{var} is 379.8 (with its own across-runs standard deviation ar- $\sigma = 96.4$) for control runs and 1,272.9 (ar- $\sigma = 465.4$) for standard. In the magenta world, the average \overline{var} is 625.31 (ar- $\sigma = 114.8$) for control runs and 1579.40 (ar- $\sigma = 123.8$) for standard. In both worlds, the p-value from a Student's t-test is under 0.05, indicating significance. Thus the quantitative results match the intuition that a significantly wider breadth of intelligent planting behaviors results when Condition 2 is met.

Discussion

The intent of this work is to set the stage for investigating the necessary conditions for open-ended evolution. By controlling for Condition 2, the experiment in this paper shows that the dynamics of Chromaria are altered significantly without the ability of Chromarians to provide new opportunities for each other. While that outcome makes sense and of course can still be tested further, the larger implication is that this experiment shows how hypotheses about the key conditions can be tested, which in turn means that the open-endedness of other alife worlds can potentially be predicted and ex-

plained. For example, any world or experiment wherein individuals do not interact (and hence do not create new opportunities for each other to meet the MC) would be expected to exhibit muted dynamics, as with the controls here. Additionally, while alife worlds often do involve interaction, Condition 3 suggests that individuals must be able to *choose* their interactions; some existing worlds thus do not meet this condition. The other conditions (once validated) similarly offer their own intriguing opportunities to assess existing alife worlds from a new perspective and thus provide insight into their own potential and limitations. While some may argue that further conditions than those proposed here are necessary (e.g. explicit competition), those can similarly be hypothesized and checked. In this way, Chromaria offers a unique opportunity to visualize the implications of different conditions, as well as opening a fascinating set of questions in its own right. For example, why does the "black ring" seem inevitable in the magenta world (Figure 6), and what might follow it far beyond 50,000 reproductions?

Conclusion

The aims of this paper were twofold: (1) to propose a new theory about what is necessary for open-ended evolutionary dynamics, and (2) to show how this theory can be tested using a new artificial life world called Chromaria. In this initial report, one hypothesized requirement for open-ended evolution was shown to be necessary; when individuals could not create new *opportunities* for each other, evolution stagnated. When the world was left unperturbed, however, it *did* exhibit some open-ended dynamics, which highlights Chromaria's utility as a platform for scientifically testing theories about evolution. Further experiments with Chromaria will test the remaining hypothesized conditions for open-ended evolution, allowing us to begin to pinpoint what is truly essential to the quest for ever-complexifying life.

References

- Bedau, M. A., McCaskill, J. S., Packard, N. H., Rasmussen, S., Adami, C., Green, D. G., Ikegami, T., Kaneko, K., and Ray, T. S. (2000). Open problems in artificial life. *Artificial Life*, 6:363–376.
- Bedau, M. A., Snyder, E., and Packard, N. H. (1998). A classification of longterm evolutionary dynamics. In *Proc. of Artificial Life VI*, pages 189–198, Cambridge, MA. MIT Press.
- Channon, A. (2003). Improving and still passing the ALife test: Component-normalised activity statistics classify evolution in geb as unbounded. In *Proc. of Artificial Life VIII*, pages 173–181, Cambridge, MA. MIT Press.
- Channon, A. (2006). Unbounded evolutionary dynamics in a system of agents that actively process and transform their environment. *Genetic Programming and Evolvable Machines*, 7(3):253–281.
- Conrad, M. and Pattee, H. (1970). Evolution experiments with an artificial ecosystem. *Journal of Theoretical Biology*, 28(3):393–409.
- De Jong, K. A. (2006). *Evolutionary Computation: A unified approach*. MIT Press, Cambridge, MA.

- Green, C. (2003-2006). SharpNEAT homepage. http:// sharpneat.sourceforge.net/.
- Holland, J. H. (1994). Echoing emergence: Objectives, rough definitions, and speculations for echo-class models. In *Complexity: Metaphors, Models and Reality*, volume XIX of *Santa Fe Institute Studies in the Science of Complexity*, pages 309–342. Addison-Wesley, Reading, MA.
- Juric, M. (1994). An anti-adaptationist approach to genetic algorithms. In *Proc. of First IEEE Conf. on Evolutionary Computation*, volume 2, pages 619–623. IEEE.
- Lehman, J. and Stanley, K. O. (2010). Revising the evolutionary computation abstraction: minimal criteria novelty search. In *Proc. of the 12th annual conf. on Genetic and evolutionary computation*, GECCO '10, pages 103–110. ACM.
- Lehman, J. and Stanley, K. O. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223.
- Lenski, R. E., Ofria, C., Pennock, R. T., and Adami, C. (2003). The evolutionary origin of complex features. *Nature*, 423:139– 144
- Maley, C. C. (1999). Four steps toward open-ended evolution. In *GECCO-99: Proc. of the Genetic and Evolutionary Computation Conf.*, pages 1336–1343. Morgan Kaufmann.
- Miconi, T. (2008). The road to everywhere: Evolution, complexity and progress in natural and artificial systems. PhD thesis, University of Birmingham.
- Miconi, T. and Channon, A. (2005). A virtual creatures model for studies in artificial evolution. In *The 2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 565–572. IEEE.
- Ofria, C. and Wilke, C. O. (2004). Avida: A software platform for research in computational evolutionary biology. *Artificial life*, 10(2):191–229.
- Orr, H. A. (2009). Fitness and its role in evolutionary genetics. *Nature Reviews Genetics*, 10:531–539.
- Pugh, J. K. and Stanley, K. O. (2013). Evolving multimodal controllers with HyperNEAT. In *Proc. of the fifteenth annual conf. on Genetic and evolutionary computation*, pages 735–742. ACM.
- Ray, T. S. (1992). An approach to the synthesis of life. In *Proc. of Artificial Life II*, pages 371–408.
- Risi, S., Lehman, J., D'Ambrosio, D. B., Hall, R., and Stanley, K. O. (2012). Combining search-based procedural content generation and social gaming in the petalz video game. In Proc. of the Artificial Intelligence and Interactive Digital Entertainment Conf. (AIIDE 2012), Menlo Park, CA. AAAI.
- Sims, K. (1994). Evolving 3D morphology and behavior by competition. *Artificial life*, 1(4):353–372.
- Spector, L., Klein, J., and Feinstein, M. (2007). Division blocks and the open-ended evolution of development, form, and behavior. In *Proc. of the 9th annual conf. on Genetic and evolutionary computation*, pages 316–323. ACM.
- Standish, R. K. (2003). Open-ended artificial evolution. *International Journal of Computational Intelligence and Applications*, 3(02):167–175.
- Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines Special Issue on Developmental Systems*, 8(2):131–162.
- Stanley, K. O., D'Ambrosio, D. B., and Gauci, J. (2009). A hypercube-based indirect encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212.
- Yaeger, L. (1994). PolyWorld: Life in a new context. *Proc. Artificial Life*, 3:263–263.

Splittable Metamorphic Carrier Robots

Tarek ABABSA
Noureddine DJEDI
LESIA Team, Computer Science Department
University Mohamed Khider of Biskra
ALGERIA

ababsatarek@yahoo.fr,n.djedi@univ-biskra.dz

Yves DUTHEN
Sylvain CUSSAT-BLANC
Vortex Team (IRIT)
University of Toulouse 1-Capitole
FRANCE

Yves.Duthen@irit.fr,sylvain.cussat-blanc@irit.fr

Abstract—Metamorphic modular robots are versatile systems composed of a set of independent modules. These modules are able to deliberately change their overall topology in order to adapt to new circumstances, perform new tasks, or recover from damage. The modules considered in this paper are cubic shapes, and we assume that each of them has a separate computational resources and it is equipped with specialized sensors to perceive the environment. In this paper, we demonstrate the ability of these robots to evolve the topology of the whole structure in order to achieve, surround and transport target objects dispersed in the environment. While performing its task, the robot may be split up in order to cope with environmental variations. Our work integrates a simplified model of biological hormone system to generate inputs for a finite-state machine (FSM) that controls the evolution process.

I. INTRODUCTION

Metamorphic self-reconfiguring modular robots are versatile systems that can change their overall shape with the intention of adapting to the task at hand. These robots are composed of a number of independent modules usually called atoms. These atoms are able to connect, disconnect one from each another, or even push/pull or exchange information and energy with the neighbor modules in order to form various structures/patterns dynamically [2,10,5]. Depending on the atoms degrees of freedom and the basic actions that can be performed in a coordinated way, several modules can perform elementary movements from position to an other position across their neighbors by changing the topology of the modules connectivity network [4,6,8,11], this action is called *robots* reconfiguration. For example, a self-configured modular robot can reform itself into a thin-linear pattern to cross a tunnel, reform into an emergency structure such as dam, shield, bridge, or even surrounding, carrying or manipulating objects.

Compared with conventional robotic systems, self-reconfigurable robots are believed to be more robust and more adaptive under dynamic environments, because on the one hand they are able to reconfigure modules to form more suited pattern with respect to the task at hand or the current situation. This property means that such robots can be survived and self repaired thanks to its ability to expel faulty modules outside the body [2,3], and in the other hand, the gathering of these modules forms a distributed system with no central controller since they are computationally independent, this property is crucial to make this kind of systems invulnerable to the failure situations or malfunction of robot modules.

Due to their many degrees of freedom, developing effective 201

control system for modular robots is recognized as one of the major challenges in the development of self-reconfigurable modular robots. These challenges attract several researchers to investigate the feasibility of providing effective solutions using the existing approaches and mechanisms, or even propose others. Some of them focused on issues of modular-robots selfreconfiguration, while others focused on issues of modularrobots locomotion. A co-evolution of both configuration and control has been also the subject of many interesting research. For example, we can cite the Karl Sims model as the first major work evolving virtual robots [1], in this work Sims used a neural network to control a morphology generated by a graphbased genotype-phenotype map so that the controller was coevolved with the morphology generator. Later, Komosinski has used this strategy again with L-System morphology generator [15] to produce artificial robots. Evert Haasdijk has used HyperNEAT to develop a reactive quadruped modular robot [16], the controllers of the individual robots act autonomously and with only local exchange of information. However, the morphology of the organism is predefined by the user. More recently, artificial Gene Regulatory Networks seem to be able to generate complex morphologies when they control a developmental system [9,12,13,14] or even to generate oscillations that give artificial creatures a mechanism to move. However, few of these works have been designed to actually take advantage of the computational power of the individual modules.

The objective of our work is to evolve the structure of a metamorphic self-reconfiguring robot to perform the task at hand by taking advantage of the computational power of the individual modules. The research presented in this paper is grounded in our previous work [7] in which we demonstrated how simple local sensing, local communication and control rules achieve useful emergent behaviors of crystalline metamorphic robots. In particular in this work, we are interested in evolving the configuration of metamorphic modular robot to transport sliding objects from their current positions to specific target positions, this process can be subdivided in three successive steps: (1) evolve dynamically the structure configuration to find and surround the objects, we use morphogen gradients to locate these objects, (2) evolve the current structure configuration to be able to transport the surrounded objects, (3) release the sliding objects whenever the final position is achieved. These steps are coded in a simple finitestate machine (FSM) that denotes all the states in which every unit of the system may be, and the possible transitions can be performed according to the required conditions, a basic

hormone system is used to control the inputs of the FSM.

II. THE CONTROL MODEL

A. The Modular Robot and its Environment

To reduce the simulation's complexity, we consider a static environment modeled with a lattice (*grid*) of 2D cells, where each of these cells may be in one of the following states: empty, occupied with a single module, occupied with an obstacle. Basically, all the modules are the same size. Each of them is controlled by the evolutionary approach shown in figure 1, perceives the environment thanks to its specialized sensors and communicates with its nearby modules [7].

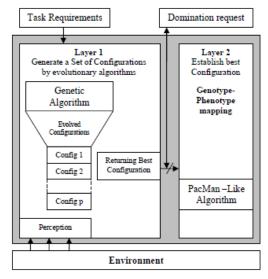


Fig. 1: Diagram of the evolutionary approach used to evolve the modular robot structure

The genetic algorithm used in this approach evolves a population of genomes that represent feasible configurations. Each of them is encoded as illustrated by figure 2.

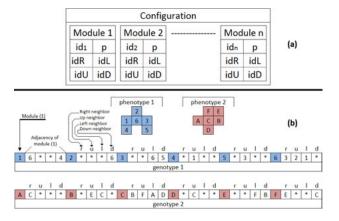


Fig. 2: Encoding a solution, (a): a genome that encod a configuration, (b): genotype-phenotype mapping

To encode an arbitrary configuration we used the adjacency matrix to denote which modules are adjacent to which other 802.

modules. Since we use Von Neumann neighborhood, a single module may have at least one adjacent module (zero adjacent is excluded) and at most four adjacent modules (respectively modules A,C in figure 2b). With this idea in mind, the adjacency matrix can be transformed into 5n size array (n is the number of modules) as illustrated in figure 3.



Fig. 3: simplified form of adjacency matrix

The *crossover* operation is applied to two different genomes that represent feasible configurations. Commonly, a point called crossover site along their length is selected, and the information after the crossover site of the two parent strings are swapped. As a result, two new children are created. However, this operation doesn't work directly with the genomes shown in figure 2. The next four additional actions are required:

- 1) From the first parent G_1 , eliminate (n-m) modules so that m < n and $\forall M_i \in G_1 / Adjacency(M_i) \neq \phi$.
- 2) From the second parent G_2 , eliminate (n-k) modules so that n=m+k and $\forall M_i \in G_2$ / $Adjacency(M_i) \neq \phi$.
- 3) From G_1 (respectively G_2), sellect two modules x_1, x_2 that have not a full adjacency list $\{\forall genome_i \mid \exists \text{ module } x_j \in genome_i \mid | Adjacency(x_j) \mid | < 4\}$ so that the translation of every module of the remaining part in G_2 by the vector $\overline{p_{x_1}p_{x_2}}$ (p_x : denotes the position of the module (x), figure 2.a) should respect the next constraint: $\forall x_j \in SubGenome_2$ $\nexists x_i \in SubGenome_1 \mid p_{x_i} = Translate \overline{p_{x_1}p_{x_2}}(p_{x_j})$ Where, $SubGenome_i$ is the remaining part of $Genome_i$.
- 4) Translate all the modules (update the p vectors) of $SubGenome_2$ by the vector $p_{x_1}p_{x_2}$, then update the adjacency list of (x_1, x_2) and integrate them together into new children genome as shown in figure 4.

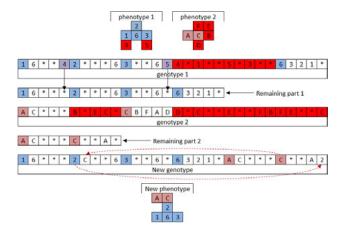


Fig. 4: The crossover operation

The *mutation* operation is applied to one genome from which a gene is chosen to be mutated. Except the part that encodes the module position, the remaining parts of the

selected gene should not be mutated. This operation proceeds as following:

- 1) Randomly select a genome G from the population.
- 2) Randomly select two genes (g_1, g_2) from the genome G so that g_2 has not a full adjacency list and the translation of the module specified by g_1 to fill a random free neighbor of the module specified by g_2 should not produce a fragmented phenotype.
- 3) Perform the translation and update the neighbor lists of (g_1, g_2) as shown in figure 5.

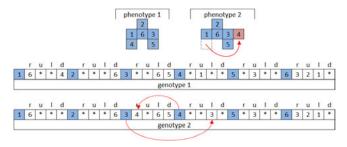


Fig. 5: The mutation operation

In this work we use the same software architecture discussed in [7], with a slight change in the sensing system (figure 6). Actually the cell-robot has no idea about the positions of the target objects, however it can sense some particular informations called *morphogens* diffused by these objects.

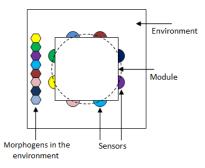


Fig. 6: Scheme of a cell-robot in an artificial environment. It has a ring of sensors (half-circles) to perceive the morphogens concentrations (hexagons)

We assume that the environment contains different morphogens. They are gradually spreading on the grid (figure 7) so that the variation on their concentrations between the neighbor cells emerges a guidance system that gives an implicit information about directions that should be followed to reach the source of these morphogens. This change improves the control model for taking advantages of not using global information and makes the system more realistic.

Our model integrates a basic diffusion algorithm to spread morphogens on each cell in the environment with respect to the following rules:

- 1) Each of the morphogens has a unique identifier.
- Each of the target objects is considered as a morphogen source, it diffuses a unique morphogen on the environment.

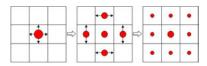


Fig. 7: Spreading a morphogen on the environment, the size of the circle represents the quantity of the morphogen, while the arrows represent the direction of spreading

- 3) The concentration of $morphogen_i$ is maximal at the position of its source (the biggest circle in figure 7).
- 4) The concentration of *morphogen*_i changes across the grid, it decreases as we move away from its source.

Algorithm 1 Spread $Morphogen_i$

```
List_A: list\ of\ empty\ cells.
List_B: list\ of\ marked\ cells.
List_A \leftarrow List_B \leftarrow \emptyset
Set to (-1) all C_{Mor_i} for each of the empty cells.
Insert the target position cell into List_A and set C_{Mor_i} to
MaxValue.
while List_A \neq \emptyset do
  if List_A.First.C_{Mor} \ge \triangle_{Mor} then Foreach(e \in Neighbor(List_A.First)) do
      if e \notin List_B then
         e.C_{Mor} \leftarrow max(e.C_{Mor}, List_A.First.C_{Mor} -
         \triangle_{Mor}).
      end if
      if e \notin List_A then
         Insert e into List<sub>A</sub>.
      end if
      EndForeach
   end if
   InsertList_A.First\ into\ List_B.
   Delete(List_A.First).
end while
```

In this algorithm:

 C_{Mor_i} : denotes the concentration of $morphogen_i$

 C_{Mor} : denotes the overall morphogens concentrations.

 \triangle_{Mor} : denotes the change in morphogen concentration through neighbor cells.

Neighbor(X): returns the list of Von Neumann neighbor cells of $Cell_X$.

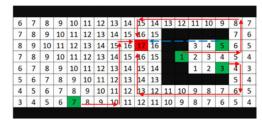


Fig. 8: Result of applying Spread $Morphogen_i$ algorithm, black squares represent obstacles, red square represents a target object, green squares represent mobile robots, values from 1 to 17 represent the concentrations of $Morphogen_i$ across the empty cells.

Figure 8 shows the result of applying *Spread Morphogen* algorithm to spread a single morphogen across all the empty cells in the environment. The green squares represent mobile robots that have a desire to reach the position of the morphogen source while the red square represents a morphogen source, so in this position the concentration of the corresponding morphogen is set to its maximum value (17 in this example). The algorithm ensures a gradient diffusion of the morphogen across each empty cell. So to find the source position, the robot will just have to follow any path that increases the morphogen concentration (red paths in figure 8).

In fact, we developed this algorithm in hope to improve our previous model [7] in which we used euclidean distance to locate target objects and to drive the system evolution (blue dashed path in figure 8). In such a model, all the units are supposed to know the exact position of all the target objects as a global information. Besides that, using euclidean distance makes the metamorphic robot unable to get out from some situations, in particular avoiding obstacles that are parabolic in shape (figure 8). To resolve this problem, we introduce f_{moving} fitness (equation 1) that should be performed by the robot to acquire morphogens as much as possible. Maximizing f_{moving} , the robot will track the morphogens concentrations from low to high level of concentration.

$$f_{moving} = \sum_{i=1}^{m} \sum_{j=1}^{n} Morphogen_i(m_j)$$
 (1)

In this equation, $Morphogen_i(m_j)$ denotes the concentration of $Morphogen_i$ perceived by the module m_j and n denote the number of the modules, m denotes the number of the morphogens sources.

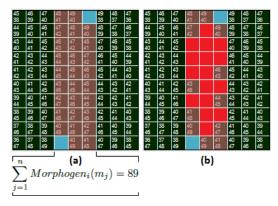


Fig. 9: The most concentrated cells \equiv equipotential area, the dark red squares represent the modular robot, the blue squares represent the target objects, values from 36 to 49 represent morphogens concentrations

Using f_{moving} as a fitness, our GA may have a tendency to converge towards local optima in which it is not defined how to sacrifice short-term fitness to gain longer-term fitness. As a result, the robot gets stuck in an equipotential area from which it can not get out anymore. This particular circumstance strongly depends on the shape of f_{moving} landscape that depends itself on the way of spreading morphogens.

In order to alleviate this problem, we introduced an activator/inhibitor coefficient (δ_i) to control the fitness f_{moving} as $_{0.04}$

illustrated in equation 2.

$$F_{Moving} = \sum_{i=1}^{m} \delta_i \sum_{j=1}^{n} Morphogen_i(m_j)$$
 (2)

In this equation δ_i is used to activate or inhibit $Morphogen_i$. Using F_{Moving} , the robot can performe the task at hand either sequentially by activating a single morphogen at a time, or in parallel by dividing the whole structure into m parts, where m>1 is the number of morphogens sources (target objects). To divide the whole structure, we used the following three rules:

- Cluster the modules into m classes, using their perceived morphogens as an input data (refer to section 3 for more details), and assign each of the modules the appropriate class-identifier.
- Each of the modules keeps the link with the sameclass modules and disconnect from the others.
- 3) For each class C_i , δ_i gets value as shown in the following equation 3, where id_{C_i} is the identifier of C_i .

$$\delta_i = \begin{cases} 1 & ifi = id_{C_i} \\ 0 & ifi \neq id_{C_i} \end{cases}$$
 (3)

Once the structure divides, each part behaves as an entirely autonomous modular robot in which only one morphogen is activated where the others are inhibited. In such a case, no equipotential area appears and each part can successfully track and surround a unique morphogen source.

B. Generate Cyclic Locomotions

The GA used in this work is basically designed for evolving the structure of modular robots as discussed in [7], it gives only the next configuration that improves the fitness at hand. However, it can be used to develop facilities for generating locomotions.

Actually, the modules can generate various motions as a combination of each module micro-movement. In particular, they are able to generate an earthworm-like locomotion as a loop of simple cyclic locomotion.

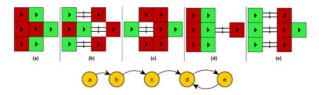


Fig. 10: Cyclic locomotion, green cells: moving modules, red cells: stopped modules

Figure 10, shows a loop of simple cyclic locomotion that can be generated using the following rules:

- (1): Define a direction for the movement.
- (2): Arrange the modules so that each of them can disconnect from its neighbor modules that are perpendicular to its direction without leaving any isolated module.
- (3): The module is able to move one step position (green modules in figure 10) once the following conditions are satisfied:
- (3.1): The neighbor cell to the direction of movement is empty.

(3.2): The disconnection from neighbor modules that are perpendicular to the direction of its movement should not leave any isolated module.

(3.3): All the module faces are contracted.

In this work, the direction of movement is defined by using the variation of morphogen concentration around the space occupied by the modules, while the modules arrangement is defined by using GA since it is a particular configuration.

C. Encapsulation into the modules

As mentioned in the introduction, we are interesting to evolve the configuration of metamorphic modular robot to transport sliding objects from their current positions to specific target positions. This process can be subdivided in three successive steps:

 $Step_1$ Track and surround the objects: the modules run a GA for evolving the whole structure in order to acquire morphogens as much as possible (using F_{Moving} as a fitness where $\delta_i = 1 \ \forall i = 1..m$). This evolution drives the modular robot towards an equipotential area in which it gets stuck and can not completely converge towards any of the target objects. (at this moment $\triangle_{F_{Moving}} = 0$).

Each module on the system can either produce two artificial hormones H^1 and H^2 (equations 4,6) or diffuse an amount of them (H^1_r, H^2_r) to control their desires to switch between the three steps.

Once the modular robot reaches an equipotential area, each of the modules starts to lose H^1 since $\triangle_{F_{Moving}} = 0$ according to equation 4. An intermodular compensation is fired (equation 7) to ensure the homogeneity of H^1 through all the modules and while $\triangle_{F_{Moving}} = 0$, H^1 keeps decreasing until it reaches a lower threshold. At this moment, a clustering method is used for determining a division scheme using the informations perceived by the modules as an input data. The whole structure is then divided into several parts where each part will be attracted by only one target object that matches with its class identifier.

 $Step_2$ Transport the objects: As a result of step (1), each target object is surrounded by several modules of the same class. Again, F_{Moving} achieves a maximum level and $\triangle_{F_{Moving}}$ converges towards 0, at this moment, every module of the class starts to lose H^2 . The modules that are in interaction with the target object lose H^2 faster than the others (φ in equation 6). Once H^2 gets lower, the modules define the direction of movement using the variation in concentration of the morphogen that denotes the final position, then a GA is called to evolve a structure that can generate a cyclic locomotion towards the defined direction.

Each substructure can push ahead the sliding object or pull it from the back while the sliding object moves from low concentration level to high concentration level of morphogen that denotes the final position. Otherwise, the modules in interaction with the sliding object stop the movement and diffuse a hormone H^3 to switch to $Step_2$ and redefine a new direction.

 $Step_3$ Release the objects: Once the sliding object arrives at the final position, it is expelled as if an obstacle or a failed module.

The dynamic of these steps can be modeled by a finite state machine as shown in figure 11, where:

Start: denotes the initial state of the system.

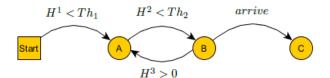


Fig. 11: FSM modeling the global task

States (A, B, C): denote respectively steps 1,2,3. (Th_1, Th_2) : denote respectively thresholds of hormones (H^1, H^2) .

In this work we integrate a highly simplified model of biological hormone system to generate inputs for FSM.

The dynamics of the hormones H^1 and H^2 are modeled as in equations 4–6:

$$H^{1} = H_{r}^{1} + \alpha_{1} \triangle_{fit} - \beta_{1} f(\triangle_{fit})$$

$$\tag{4}$$

$$f(x) = \begin{cases} 0 & if x \neq 0 \\ 1 & if x = 0 \end{cases}$$
 (5)

$$H^{2} = H_{r}^{2} + \alpha_{2} \triangle_{fit} - \varphi \beta_{2} f(\triangle_{fit}) \tag{6}$$

Where, $\triangle_{fit} = |F(t) - F(t-1)|$: denotes the change of the fitness over the time. (α_1, α_2) : are two coefficients used to accelerate the production of H^1 and H^2 . (H^1_r, H^2_r) : represent the received hormones. (β_1, β_2) : are two coefficients used to decelerate the production of H^1 and H^2 . φ : is a coefficient used to enhance the deceleration of producing H^2 .

The function $D_{x,y}^i(t)$ models the diffusion of the hormone H^i at time t as described in the following equation 7, where d_i represents the diffusion coefficient of H^i :

$$D_{x,y}^{i}(t) = d_{i}|x - y|H^{i}(t)$$
(7)

III. CLUSTERING THE MODULES

Clustering is the task of finding natural groupings among objects in such a way that objects in the same group (called a cluster) share similar values [18]. In our work we use clustering for partitioning modules into several groups so that the whole structure of the modular robot will be split up in order to cope with environmental variations. From this point of view, the perceived information of each module is considered as a data point.

In this section we discuss three clustering methods in the aim of choosing the more efficient of them for our study. These methods are applied to the same data inputs.

A. K-means Clustering

K-means is a well known algorithm commonly used to solve clustering problems, it is based on minimizing the overall sum of the squared errors between each pattern and the corresponding cluster center. This can be written as minimization of the following objective function:

$$E = \sum_{i=1}^{K} \sum_{x \in C_i} ||x - m_i||^2$$
 (8)

K-means clustering proceeds as shown in algorithm-2, where m_i represents the center of the cluster C_i . This algorithm converges when there is no further change in assignment of

Algorithm 2 K-means

while $\exists \triangle m_i \neq 0$ do

- (1): Initialize the k cluster centers.
- (2): Assign each input data point to one of the existing clusters according to the closest Euclidean distance from the clusters.
- (3): Compute the mean of each cluster to update its center. **end while**

instances to clusters. In such situation, all the centers m_i are stable in values ($\triangle m_i = 0$).

Depending on the first initialization of the clusters, k-means algorithm converges in 3–8 iterations, and gives a good data-classification quality (a low overall error).

B. GA-Based Clustering

The goal of using GA to solve clustering problem is to enhance the quality of clusters. In fact the GA works with a population of feasible solutions called genomes, each genome encodes a solution and is assigned a fitness value to describe how good solution it represents.

Starting from a random initial population, the GA stochastically selects a set of individuals (based on their fitness) and modifies their genetic codes by means of operators: mutation and crossover to form a next evolved population.

The genome that encodes the clusters is shown in figure 12, where id represents the data point identifier and c represents its corresponding cluster.

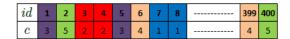


Fig. 12: Genome that encodes the clusters

A feasible solution assigns each element of the data point to a cluster. The center of each cluster is determined and the overall error E of the solution is calculated using equation 8. The fitness should be inversely proportional to the overall error of the corresponding genome, in this paper we adopt a basic form shown in equation 9.

$$Fitness = \frac{1}{\sqrt{E}} \tag{9}$$

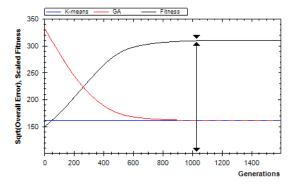


Fig. 13: GA-Clustering

As shown in figure 13, the GA improves the quality of the clusters in each generation. However it seems to be of little interest for our work since it needs an average of 1025 generations to converge around k-means solution.

C. SOM-Based Clustering

Kohonen Self Organizing Map, or SOM, is an artificial neural network based on an issue of unsupervised learning with the aim of mapping a high-dimensional data to a low-dimensional space (figure 14) which is formed by arranging the computational neurons into a grid [17,19].

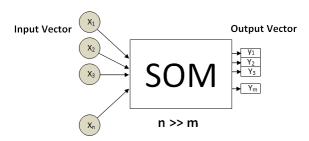


Fig. 14: Global sheme for SOM clustering

The way that SOM goes about organizing itself describes a self organization process, during this process a competition between neurons is invoked, and each neuron is allowed to change its weight vector to become more like samples in hopes to win the next competition. The general steps of the learning process are described in algorithm 3:

Algorithm 3 SOM

(1)-Initialization:

Randomly initialize the weight vectors of the map

(2)-Begin training:

while $t \le 1$ do

An input weight vector is applied to the map

The node most like the input is selected (Best Matching Unit).

The weight vectors of BMU and surrounding neighborhood nodes are scaled towards the input.

The Learning rate and the ray of neighborhood are decreased.

end while

(3)-Apply input vector:

The BMU now is the node most like that provided.

In first stage, the best matching unit (BMU) should be selected, usually as the unit matching to the shortest euclidean distance (equation 10) between the input vector and the units of the map.

$$D_{(v_2,v_1)} = \sqrt{\sum_{i=1}^{n} (v_{2i} - v_{1i})^2}$$
 (10)

Next, the neighboring weights should be scaled, so firstly we should select the units considered as neighbors to the winner unit, then we should determine how much each weight can become more like the input vector. The neighbors of a winning unit can be determined using a number of different methods.

In this paper we opted to use a gaussian function (equation 11) where every point with a value above zero is considered as a neighbor.

$$G(x,y) = \alpha e^{-\frac{x^2 + y^2}{\beta}} \tag{11}$$

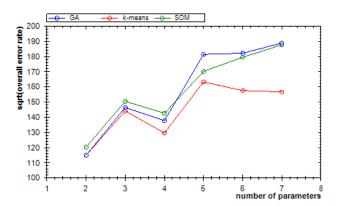


Fig. 15: Overall error of k-means, SOM, GA clustering methods

To compare these clustering methods, a data set is captured from a random robot configuration. The captured informations represent the morphogen diffusion across the cells occupied by the robot modules.

As shown in figure 15, we note that k-means is not just the easiest algorithm to be realized, but also our natural choice since it gives a high quality of data clustering (for the purpose of our work) and has low computation cost (converges in 3 to 8 iterations).

IV. RESULTS

In this experiment, the environment is modeled as a 2D grid composed of 25x10 cells as shown in figure 17, where, the shaded cells represent obstacles, the blue cells represent sliding objects, and the 16 unit modular robot is represented by the red cells. Each of the sliding objects produces a unique morphogen that is spreaded gradually on the environment. The mission should be performed by the modular robot is to track the sliding objects that are randomly dispersed in the environment and transport them into a predefined final position (we assume that the final position diffuse a unique morphogen called $Morph_{Final}$).

The following parameters are used to set up the simulation: Max morphogen concentration = 50, hormone accelerators $(\alpha_1,\alpha_2)=(0.8,0.8)$, hormone decelerator $(\beta_1,\beta_2)=(0.3,0.3)$, hormone diffusion coefficient d_i =0.5, the cumulation of any hormon H^i can not exceed 10 (otherwise the additional value is ignored except for the non divided structure). This setup has been empirically determined, through a set of tests.

During the convergence of the genetic algorithm, it is interesting to observe the evolution of the structure towards the best solution. As it is shown in figure 17(a.b), the modular robot starts form its initial position and evolves its structure to aquire morphogens as much as possible. As a result, the modular robot converges more and more towards the most concentrated cells (red-brick cells in figure 17(a.b.c)), at this 807

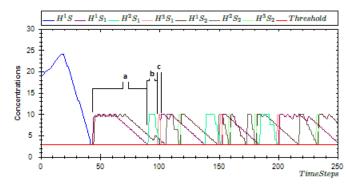


Fig. 16: Hormones concentrations during the time of simulation, (a,b,c): parts of the global task

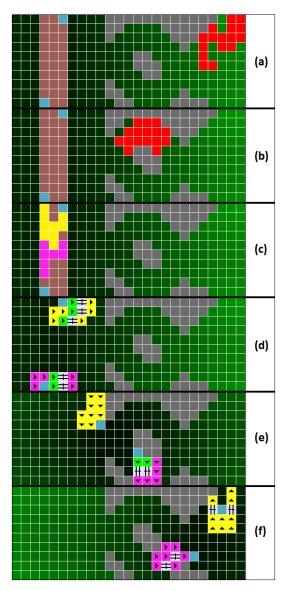


Fig. 17: The modular robot during the evolution

moment, the structure is not yet divided, and only H^1 is produced. Once the structure reaches the equipotential area (figure 17c), $\triangle_{F_{Moving}}$ approaches to 0 and H^1 starts to decrease (blue curve in figure 16).

When the H^1 concentration gets down under the threshold 2.5 (empirically determined), the structure is believed to be steady for a long time and is ready to be divided into several parts. A k-means algorithm is called to cluster the dataset (informations perceived by the modules) and to assign each module to a class preparing to the separation process. Next, each module tests its neighbors and disconnects from those that have not the same class identifier. As a result, the modules are separated in two classes and the structure is divided in two parts (figure 17c), where these parts should not be divided anymore and each of them converges only to the object attracted with, then it surrounds this object and evolves its configuration to be able to perform a cyclic locomotion to the direction by which the concentation of $Morph_{Final}$ is being increased (subsript b in figure 16), otherwise, a hormone H^3 is diffused to evolve an other configuration to move toward a new direction. As a result, the sliding object gets closer to the final position.

The global system dynamic is already modeled by a FSM that is encapsulated in every module to switch between steps (parts of the global task) a,b,c in figure 16 while the hormon system produces H^1 , H^2 , H^3 to generate inputs to this FSM. Coupling between the hormone system and the FSM is illustrated as following:

 $(State = Start \land H^1 < 0.3) \mapsto (State \leftarrow A, H^2 \leftarrow Max)$ $(State = A \land H^2 < 0.3) \mapsto (State \leftarrow B)$

 $(State = B \land H^3 > 0) \mapsto (State \leftarrow A, H^2 \leftarrow Max)$

Observing these rules, we notice that the second and the third rules, create such an interesting cycle that can be used to create a generalized process for more complex tasks.

V. CONCLUSION

In this paper, we presented a decentralized approach that evolves the ability of metamorphic robots to perform the task at hand. This approach is based on our previous work [7] in which we used a GA coupled with a PackMan-like algorithm for evolving the structure of a modular robot. In this study the genetic algorithm is used to generate the next better configuration of the modular robot, while the PackMan-like algorithm is used to drive the self-reconfiguration process. Switching between generating better configuration and reforming to the new configuration emerges an adaptive locomotion for the modular robot.

A more complex task is considered in this work, and a new evolutionary approach is presented.

The first improvement we can talk about is the ability of our approach to drive the modular robot into the target objects without being stuck by obstacles that are parabolic in shape (figure 8). In fact, using a gradient of morphogens instead of euclidean distance is not just a natural choice since the modules are not supposed to have global informations but it gets also a significant improvement to the quality of objects-tracking in our system.

The experiment presented in this paper shows the capacity of the artificial hormone system to control the finite state machine (FSM) that schedules the steps to perform the global task. To do that, the global system task dynamic is modeled 0.08

by a FSM, and then an artificial hormone system is used to control transitions between the FSM states. As a result, the robot's behavior is controlled by the hormone system.

An other interesting property of our approch is the ability to generate a separation strategy for the modules according to some circumstances. It should be interesting to investigate the feasibility of biological cell division techniques as well as the multi-objective techniques for generating such strategy.

REFERENCES

- K. Sims. Evolving 3d morphology and behavior by competition. Artificial Life IV, pages 28-39, 1994.
- [2] U. P. Schultz, M. Bordignon, K. Stoy, Robust and Reversible Self-Reconfiguration. The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, USA, October 2009.
- [3] D. Christensen. Experiments on Fault-Tolerant Self-Reconfiguration and Emergent Self-Repair. In IEEE Symposium on Artificial Life (AL-IFE'07), pages 355-361, 2007.
- [4] D. Christensen, A Unified Simulator for Self-Reconfigurable Robots, In 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, pp. 22-26, 2008.
- [5] M. Yim, W. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. Chirikjian. Modular self-reconfigurable robot systems [grand challenges of robotics]. IEEE Robotics and Automation Magazine, 14(1):43-52, 2007.
- [6] S. Vassilvitskii, J. Kubica, E. Rieffel, J. Suh, M. Yim. On the General Reconfiguration Problem for Expanding Cube Style Modular Robots. In IEEE Intl. Conf. on Robotics and Automation (ICRA), 2002.
- [7] T. Ababsa, N. Djedi, Y. Duthen, S. Cussat-Blanc. Dencentralized Approach to Evolve the Structure of Metamorphic Robots (regular paper). In: IEEE Symposium on Artificial Life (IEEE-ALife 2013), april 2013.
- [8] M. Vona, D. Rus. A Physical Implementation of the Self-reconfigurable Crystalline Robot. In Proc. of the IEEE Intl Conf. on Robotics and Automation 2000, San Francisco, CA, Apr. 24-28, 2000, p. 1726-1733.
- [9] J. Bongard and R. Pfeifer. Evolving complete agents using artificial ontogeny. Morpho-functional machines: The new species (designing embodied intelligence), pages 237-258, 2003.
- [10] A.C. Van Rossum, J. H. Van Den Herik, Designing Robotic Metamorphosis. In Proc. of the 22nd Benelux Conference on Artificial Intelligence (BNAIC-2010), Luxembourg, 2010.
- [11] G. S. Chirikjian, Kinematics of a metamorphic robotic system. In IEEE International Conference on Robotics and Automation Proceedings, Volume 1, pp. 449-455, 1994.
- [12] L. Schramm, Y. Jin, and B. Sendho. Emerged coupling of motor control and morphological development in evolution of multi-cellular animats. Advances in Artificial Life: Darwin Meets von Neumann, pages 27-34, 2011
- [13] Sylvain Cussat-Blanc, Jordan Pollack. A Cell-based Developmental Model to Generate Robot Morphologies (regular paper). In: Genetic and Evolutionary Computation COnference (GECCO 2013).
- [14] Meng, Y, Zheng, Y and Jin, Y. A Morphogenetic Approach to Self-Reconfigurable Modular Robots using a Hybrid Hierarchical Gene Regulatory Network. On Artificial Life XII, 2010.
- [15] Komosinski M, Rotaru-Varga A. From directed to openended evolution in a complex simulation model. In: Bedau MA, McCaskill JS, Packard NH, Rasmussen S, editors. Artificial Life 7. Cambridge, MA: The MIT Press, 2000. p. 2939.
- [16] Haasdijk, E., Rusu, A.A., Eiben, A.: Hyperneat for locomotion control in modular robots. In: 9th International Conference on Evolvable Syste ms (ICES 2010). pp. 169180 (2010)
- [17] Balakrishnan, P. V., M. C. Cooper, V.S. Jacob, P.A. Lewis. A Study of the Classification Capabilities of Neural Networks using Unsupervised Learning, Psychometrika 59(4): 509-525 (1994).
- [18] Jain, A.K., M.N. Murty, P. Flynn. Data Clustering: A review. ACM Computing Surveys 31(3): 264-323. (1999)
- [19] K.Mehrotra, C. Mohan, and S. Ranka. Elements of Artificial Neural Networks. MIT Press, (1996)

The Effect of the Network Structure Differences on the Diffusion of Items

Shota Onoda^{1,2}, Shohei Kato^{1,*} and Atsuko Mutoh¹

Dept. of Computer Science and Engineering, Graduate School of Engineering, Nagoya Institute of Technology, Gokiso-cho Showa-ku Nagoya 466-8555 Japan
 NGK Insulators, LTD., 2-56 Suda-cho, Mizuho, Nagoya 467-8530 Japan
 * shohey@nitech.ac.jp

Abstract

This paper presents a multiagent-based simulation approach to the effect of network structure difference on the diffusion of items. Recently, the rapid spread of information and communication technology induces multiplexing of our communication space. As a result, diffusion of products in the market has been changed because the communication affects our behaviors and state of own mind. Network externality is an effect that value of a product depends on the market penetration. It has been known that markets of products having network externality are greatly influenced by interpersonal communication. In this paper, we constructed two network, "offline network" and "online network" that refer to networks before and after the development of information and communication technologies, focusing on such changes and analyzed the differences between network structures. We verified the effect that difference of network structures affects the diffusion of items in network effect markets. We discussed the property of diffusion process in each network and how easily monopolistic diffusion can occur depending on the network structures.

Introduction

Recently, a socializing method and other parties of association have been changed because of the development of information and communication technologies such as the Internet and mobile phones. For example, mobile phones make it possible to communicate with anyone, anytime, and the Internet and Social Network Service (SNS) make it easy for people who have something in common to communicate through online communities. It is thought that communication and exchanging information with others, and their behavior, affects our own actions and psychology. The network effect (also called network externality or demand-side economies of scale) in markets is an example of interaction with others producing a major effect on individuals (Katz and Shapiro, 1985). Briefly, the network effect means that as the number of users an item has increases, that item becomes more attractive to others. In this situation, it is thought that the diffusion rate of items within one group of friends affects the decision about whether or not to buy something. When two similar items are in a race to be the most popular, it is common for only one item to be shared exclusively

because of positive feedback and items selling faster if they are already doing well. This phenomenon is called "winner-takes-all" by Arthur (1996).

There has been much previous research about the network effect (e.g., Ozawa and Nakayama (2010); Weitzel et al. (2003)). Kaneko et al. (2006, 2005) constructed a model that considered the asymmetricity of information and analyzed customers' purchase decision-making processes. In the model each consumer had different information with respect to others' purchase behavior, and they reported that the market becomes inefficient if customers are unaware of each other's behavior. Kawamura and Ohuchi (2005) analyzed the effectiveness of the present strategy, in which businesses provide their services without charge. They examined the effectiveness of two present strategies: a simple present strategy and friend present strategy, and discussed the effectiveness of present strategies in different network structures. Iba et al. (2001) analyzed the format competition of video cassette recorders, that is to say a typical standard race interaction between customers, by the artificial market model with multiagent approach. The simulation observed the emergence of locality, which is caused by the local influence, and the results showed that the local clusters provide the brakes on the winner-take-all phenomenon. Mizutani (2002) proposed a simulation model with evincive individual relationships. He used network structures that describe the relationship in urban and provincial areas, and showed that differences to these structures can affect the diffusion

In a network effect market, the situation at the early stage is significant in terms of the final diffusion result (Liebowits and Margolis, 1994). Therefore, we need to examine the early stage and come up with a definition for the customer group that decides to buy an item at the early stage and how the diffusion process proceeds from that point. Additionally, it seems to be important to study acquaintance network structures that show the relationships between customers.

We propose a model that draws on Mizutani (2002)'s model that focuses on the relationship between network structure differences and the diffusion state. First, we de-

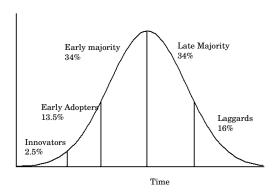


Figure 1: Innovator theory: five types of customers to adopt a new item and their population distribution.

fine the customer group that decides to buy an item at an early stage and add conservative buying to the purchase model for realistic simulation. In the real world, it is well-known that buying patterns are related to differences in a customer's character (Rogers, 2010). Therefore, we constructed an agent model that considers innovator theory and classifies customers into five categories in order to analyze the acquaintance networks in each category. We also added conservative buying in the purchase model in order to make it possible to analyze realistic diffusion process. We show the differences of acquaintance networks that each adopting category has as well as network structure differences, and then discuss how these differences the affect the diffusion and the process of the diffusion of items.

Innovator theory

Innovator theory is a sociological theory first proposed by Rogers (2010). In this theory, customers are classified into five categories (see Figure 1) on the basis of how quickly they develop a purchasing attitude toward new items.

• Innovators: 2.5% of all customers.

• Early adopters: 13.5% of all customers.

• Early majority: 34% of all customers.

• Late majority: 34% of all customers.

• Laggards: 16% of all customers.

The character of the customers in each of the categories is also analyzed. Early adopters tend to have a higher income, intelligence, a friendlier attitude toward science, and a more reasonable personality than late adopters.

Proposed Model

In this paper, we assume that if a new item, or two conflicting new items, appears in the market, there will be a network effect. If two new items exist in the same market, customers can choose whichever item is preferable to them. We assume that customers do not know the diffusion rate of item in an entire market; they only know the local popularity in their own acquaintance network.

In this section, we define an agent as a customer, and a network as both an acquaintance relationship network and a purchase action model.

Agent Model

We define agents as customers who exist in society. An agent is comprised of adopting an item $(adopting_i)$, social attribute (AT_i) , and its two derivative attributes: social position (SP_i) and threshold recognizing desire to buy (T_i) .

$$agent_i(adopting_i, AT_i, SP_i, T_i).$$
 (1)

 $adopting_i$ describes three values: adopting item A or B, or not adopting. Agents can have only one item, and although they are allowed to replace on item with another, they cannot have two items at the same time.

$$adopting_i = 0 \text{ or } 1 \text{ or } 2.$$
 (2)

Social attribute describes an agent's social position, e.g., age, occupation, educational background, and so on. Elements of the social attribute at_{ik} describe the real value from 0 to 1 randomly.

$$AT_i = (at_{i1}, \dots, at_{ik}, \dots, at_{iK}),$$

$$(1 < k < K).$$
(3)

Social position is calculated by the following equation using the social attribute.

$$SP_i = \sqrt{\sum_{k=1}^K at_{ik}^2}. (4)$$

In the innovator theory, there is a positive correlation between the height of the social attribute and the speed of the purchase time. The agents are classified into five categories according to value of SP_i .

"Threshold recognizing the desire to buy" describes the speed of a purchasing attitude to a new item. Agents will decide to buy an item if the local popularity among their acquaintances is over their own threshold. Agents who have a low threshold have the desire to buy a new item in an early stage easily. On the other hand, agents who have high threshold get this desire in the late stage. They are cautious

about buying new items, they do not buy any until the majority of their acquaintances have. The threshold recognizing the desire to buy is indicated by the following equation using the social position.

$$T_i = 0.5 - \frac{SP_i}{2\max SP},\tag{5}$$

where maxSP means the max value of SP_i .

Network Model

The section describes a network model with an agent as a node and acquaintance relationships as links. Mizutani (2002)'s research included two different types of structures, strong network and weak network, based on the similarity of social attributes between people, namely homophily. Homophily refers to people making a relationship with someone and the giving priority to people with whom the social distance between them is close. Social distance is indicated by the social attribute. In the case of using several social attributes, they are treated as dimensions.

In the past, before the recent development of information and communication technologies, it was not as easy to contact others as it is now, so geography became a key factor in acquaintance relationships. However, these days it is possible to contact and communicate with people who have at least one thing in common with oneself, so homophily is getting weaker than before. In this study, we consider two networks, "offline network" and "online network", based on the homophily in Mizutani (2002). "Offline network" and "online network" refer to networks before and after the development of information and communication technologies.

• Offline Network

This is the network before the development of information and communication technologies. Homophily has a significant influence and links are made between people with a similar social attribute in all dimensions, that is, birds of a feather. The offline network has an acquaintance relationship with strong inbreeding and is created by using all social attributes. Social distance $(padis_{ij})$, which is defined by using Euclidean distance in all social attributes, is indicated by the following equation.

$$padis_{ij} = \sqrt{\sum_{k=1}^{K} (at_{ik} - at_{jk})^2}.$$
 (6)

• Online Network

This is the network after the development of information and communication technologies. Homophily has a weaker influence and links are made between people with a similar social attribute in as little as one dimension. Heterophily is superior to homophily in this network.

The online network has an acquaintance relationship like weak-ties and is created by using at least one social attribute. First, we calculate the difference of each social attribute's element between $agent_i$ and $agent_j$, and next, we use the following equation to determine their minimum value of the difference $(prdis_{ij})$.

$$prdis_{ij} = \underset{k=1,\dots,K}{\operatorname{argmin}} (|at_{ik} - at_{jk}|). \tag{7}$$

In this paper, each network has same number (L) of links that exist in society fixedly in order to compare the effect that different network structures have on the diffusion of items. Links between $agent_i$ and $agent_j$ are composed in ascending order according to $padis_{ij}$ in all combinations of agent pairs. Clustering coefficient defined by Watts and Strogatz (1998) is well known as a measure of the degree to which people in a network tend to cluster together. The clustering coefficient of offline and online network was 0.404 and 0.046, respectively.

Purchase Action Model

We proposed a purchase action model based on the Engel-Blackwell-Miniard (EBM) model by Engel et al. (1994). In the EBM model, a customer's decision-making process to buy items includes the following sequential phases: "recognize desire," "collect information," "assess alternative before buying," "buy," "consumption," "assess alternative after buying," and "dispose." In this paper, we do not touch on "consumption," "assess alternative after buying," or "dispose" because our model focuses only on making the decision to buy items. Customers recognize their desire to buy something after collecting information because we assume that customers can only collect information about items their own acquaintances have purchased. Therefore, we define the process of a customer's decision making as a sequence of "collect information," "recognize desire," "assume alternative before buying," and "buy." Figure 2 shows the flowchart of the purchase action by a single agent.

Agents in the innovator category are customers who buy new items at the first stage, so we assumed that they make the decision to buy items not on the basis of the local popularity in their acquaintances but rather on their own judgment. Therefore, in this paper, we propose that innovators have a particular purchase action model if they have not used any items yet.

In this section, we first the innovator's particular purchase action model and then define each phase of the other purchase action model.

Innovator's Purchase Action Model

Innovators are the earliest customers to buy new items, so we define them as agents who have the desire to buy new items as soon as they appear on the market. We assume that they make the decision to buy new items themselves. We therefore define probability p[0,1] as the height of their assessment of item A. They buy item A with probability p and buy item B with probability 1-p. When they trade up

TC 1 1	4	A . •	D .		TD 1 1
Table	١.	Action-	-I)ecu	cion	Table

Using item	$N_a > N_b$	$N_b > N_a$	$N_a = N_b$
Item A	Hold item A	Trade up item B	Hold item A
Item B	Trade up item A	Hold item B	Hold item B
Non-adopting	Buy item A	Buy item B	Hold off

for another, they follow the same as purchase action model of the other categories.

"Collect Information" Phase

In this phase, agents collect information about who has which items among their acquaintance agents. We define Na as the number of users who own item A, Nb as the number of users who own item B, and Nn as the number of agents who are not having any items.

"Recognize Desire" Phase

In this phase, agents recognize desire when they meet the following equation.

$$\frac{Na}{Na+Nb+Nn} > T_i \ or \ \frac{Nb}{Na+Nb+Nn} > T_i. \ \ (8)$$

If the agent does not recognize desire, the purchase action will finish.

"Assess Alternatives Before Buying" Phase

In this phase, agents recognize desire for an alternative item before buying and judge which they prefer by the following equation.

- Na > Nb: Item A is preferable.
- Nb > Na: Item B is preferable.
- Na = Nb: can not judge which item is preferable.

"Buy" Phase

Agents who have assessed alternatives are now ready to buy. They follow the action-decision table shown in Table 1 by a combination of judgment in an earlier phase and own item Agents will buy an item that is preferable if they have not used any yet. If an agent already has an item, the agent will trade up for another one if it is considered preferable.

Simulation

We performed simulations with the networks using the following parameters. Number of agent N=2500, number of links L=25000, and number of social attribute elements K=5. No agents had any items at the default position. Agents were randomly selected one by one and allowed to perform a purchase action process. A step was finished when all agents were selected. Each agent could only perform one purchase action per step. The simulation was finished when all agents did not buy or trade items in a step, meaning the market had become convergent.

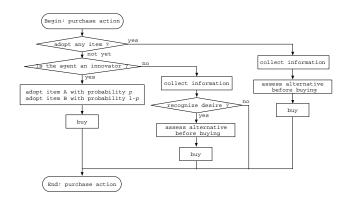


Figure 2: Flowchart of purchase action by a single agent.

First, we show the difference between the two networks. Next, we discuss the diffusion process in the case of one item in the market in simulation 1. Finally, we assume that two items are entering a diffusion race and discuss how often monopolistic occurs during the process in that situation.

Difference of Network Structures

First, we analyze the difference between the structures of the offline and online networks. We have constructed ten offline and online networks respectively and measured the distribution of which category each agent's acquaintances belong to. The results are shown in Table 2 and 3. Different agents in each adopter category have acquaintance relationships with agents in particular categories depending on the networks. In the offline network, an agent has an acquaintance relationship with an agent who has a similar buying attitude because there are many acquaintances between agents in the same category. In contrast, in the online network each agent has acquaintances with agents belonging to several categories. The number of common acquaintance that exists between two agents who have an acquaintance relationship with each other is also different. We counted the number of common acquaintances into linked agent pairs for 10,000 sample links extracted randomly from offline and online networks. As a result, the average number of common acquaintances in offline and online networks was 7.2 and 1.7, respectively. If there are many common acquaintances in a network, the transmission efficiency of the information is low because this network has a loop structure. It is in easily formed small groups that the network effect works both strongly and locally. In contrast, if there are relatively few common acquaintances in a network with the exception of having ring or linear structure, the transmission efficiency of information is high because the network has a hub structure and it is difficult to form a small group.

Simulation 1: The Diffusion Process for One Item

First, we assume that only one new item, A, appears in

Table 2: Acquaintance Relationship of Offline Network

	Innovators	Early Adopters	Early Majority	Late Majority	Laggards	Average # of acquaintance
Innovators	4.3 (35.2%)	7.7 (63.2%)	0.2 (1.6%)	0	0	12.2
Early Adopters	1.4 (8.2%)	9.8 (57.5%)	5.8 (34.3%)	0	0	17.0
Early Majority	0	2.3 (11.3%)	13.6 (67.0%)	4.4 (21.7%)	0	20.3
Late						
Majority	0	0	4.3 (20.5%)	14.5 (67.5%)	2.6 (12.0%)	21.4
Laggards	0	0	0	5.5 (27.4%)	14.4 (72.6%)	19.9

Table 3: Acquaintance Relationship of Online Network

	Innovators	Early	Early	Late	Laggards	Average # of
		Adopters	Majority	Majority		acquaintance
Innovators	2.2 (10.9%)	7.2 (35.8%)	7.9 (39.1%)	2.7 (13.1%)	0.2 (1.1%)	20.2
Early						
Adopters	1.4 (6.7%)	5.2 (26.4%)	7.9 (40.1%)	4.4 (22.3%)	0.9 (4.5%)	19.8
Early						
Majority	0.6 (2.9%)	3.2 (15.6%)	7.8 (38.9%)	6.5 (32.0%)	2.1 (10.6%)	20.2
Late						
Majority	0.2 (1.0%)	1.7 (8.7%)	6.4 (32.2%)	7.7 (38.1%)	4.0 (20.0%)	20.0
Laggards	1.0 (0.2%)	0.7 (3.6%)	4.6 (22.9%)	8.5 (42.5%)	6.1 (30.8%)	19.9

the market, and we study the effect that network structure differences have on the diffusion process of this item. We set a probability that an innovator buying item A is equal to 1. The number of buyers in each step is shown in Figure 3, and the process of diffusion rate is shown in Figure 4. There is a clear difference in the diffusion processes. The online network's diffusion speed was slower than that in offline network at the early stage, but it increased quickly after the diffusion rate was over 15% (375 users).

The offline network had acquaintance relationships between agent pairs belonging to the same category and a lot of common acquaintances, and it was easy to form small groups that the network effect works both strongly and locally. Therefore, the diffusion moved ahead steadily. However, the diffusion speed into the entire market was slow because of poor transmission efficiency.

In contrast, the online network had acquaintance relationships between each agent and agents belonging to several categories and few common acquaintances, and it was difficult to form small groups. Therefore, the diffusion proceeded slowly at the early stage but spread into the entire market quickly after diffusion rate was over 15% because its structure resulted in a good transmission efficiency.

Simulation 2: Monopolistic Diffusion

Next, we assume that two contrasting items, A and B, are launched at the same time and are engaged in a diffusion race to get shares, and we study the effect of the network structure differences on the diffusion race. We looked into the relationship between probability p with which an innovator buys item A and the occurrence rate of monopolistic diffusion such that item A gets more than 90% share. We have conducted 100 simulation runs with every p from 0.25 to

0.75 at intervals of 0.05, and checked whether monopolistic diffusion occurred or not. The occurrence rate of monopolistic diffusion is shown in Figure 5. The results indicated that the online network had monopolistic diffusion more easily than that in offline network.

And we also looked into the diffusion process when monopolistic occurs. In the case of p=0.5, we chose a result in which one item got a monopolistic share as a sample, the process of diffusion rate is shown in Figure 6. We assume that the majority as winner, the minority as loser in this result. In the offline network, minorities were left in small groups even though in such situation because it was easy to construct small groups that the network effect works both strongly and locally. In contrast, in the online network, the minority finally disappeared because it was difficult to construct small groups. Therefore, the online network is had monopolistic diffusion more easily than that in offline network.

Conclusion

We propose a model for describing the diffusion process. This model classifies agent into five categories on the basis of the speed of making a decision to buy a new item and analyzes the network structures belonging to the agent of each category. And we added conservative buying to the purchase model and analyzed the diffusion process, which Mizutani's model is not able to do. Next, we simulated two situations: diffusion of one item, and two items that enter a diffusion race. Results highlighted the differences in the diffusion process, the time needed to diffuse an entire market, and how easily monopolistic diffusion can occur depending on network structures.

It is a fact that human relationships are complicated. The

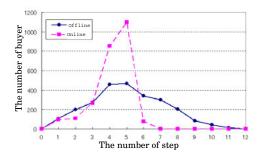


Figure 3: Number of buyer in each step.

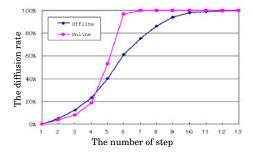


Figure 4: Process of diffusion rate.

model that we proposed in this paper only uses one of many possible factors, so our result does not describe a real diffusion process exactly but rather a potential one. We intend to develop a more realistic model as our future work.

Acknowledgment

This work was supported in part by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research under grant #25280100 and #25540146.

References

- Arthur, W. B. (1996). Increasing returns and the new world of business. *Harvard Business Review*, 74(4):100–109. July-Aug.
- Engel, J. F., Blackwell, R. D., and Miniard, P. W. (1994). Consumer Behavior 8th Edition. The Dryden Press Series in Marketing.
- Iba, T., Takenaka, H., and Takehuji, Y. (2001). Reappearance of Video Cassette Format Competition Using Artificial Market Simulation. *Transactions of Information Processing Society* of Japan, 42(SIG14(TOM5)):73–89. (In Japanese).
- Kaneko, Y., Nishino, N., Oda, S. H., and Ueda, K. (2005). Decision Making with Incomplete Information and Network Externality in Product Market. *Proceedings of the Japan Society of Mechanical Engineers*, 15:156–159. (In Japanese).
- Kaneko, Y., Nishino, N., Oda, S. H., and Ueda, K. (2006). Purchase Behavior under Asymmetric Information in Product Market with Network Externalities. *Transactions of Information Procesing Society of Japan*, 47(5):1473–1482. (In Japanese).

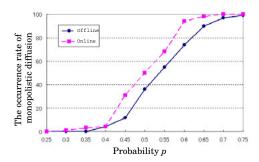


Figure 5: Monopolistic diffusion.

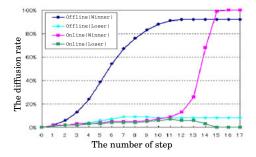


Figure 6: Process of diffusion rate.

- Katz, M. L. and Shapiro, C. (1985). Network Externalities, Competition, and Compatibility. American Economic Reviews, 75:424–440.
- Kawamura, H. and Ohuchi, A. (2005). Evaluation of Present Strategies in Multiagent Product Market Model with Network Externality. *Transactions of the Operations Research Society* of Japan, 48:48–65. (In Japanese).
- Liebowits, S. J. and Margolis, S. E. (1994). Network externality: an uncommon tragedy. *Journal of Economics Perspectives*, 8(2):133–150.
- Mizutani, N. (2002). The Effect of Interpersonal Communication on the Competitive Diffusion of New Products. *The Transactions of the Institute of Electronics, Information and Communication Engineers D-I*, J85-D-I:582–591. (In Japanese).
- Ozawa, J. and Nakayama, Y. (2010). Efficiency of Present Strategies in Acquiring a Larger Customer Share in a Market with Network Externalities: an analysis of a small world network model. *Journal of Japan Society for Fuzzy Theory and Intelligent Informatics*, 22(2):178–189. (In Japanese).
- Rogers, E. M. (2010). *Diffusion of Innovations, 4th Edition*. Simon and Schuster.
- Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, 393(4):440–442.
- Weitzel, T., Beimborn, D., and Konig, W. (2003). An Individual View on Cooperation Networks. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, pages 10–pp.

An Artificial Chemistry for the 'Lipid World' Scenario

Barak Shenhav

Department of Software Engineering, Afeka - Academic College of Engineering, Tel aviv, Israel baraks@afeka.ac.il

Abstract

'Metabolism first' scenarios are amply suggested as an alternative or precursor for the emergence of the 'RNA world' that may have prevailed before contemporary life as we know it. The 'Lipid world' scenario advocates 'metabolism first'. The commonly used computational model for the scenario (the GARD model) emphasizes that reaction catalysis is not a binary phenomenon, but rather a matter of magnitude. Simulations of the model revealed lack (or at most very low capacity) to undergo natural selection. Here, it is proposed that this cavity emerges from the underlying distribution of catalytic values typically employed by the model. In particular, it is shown that the propensity of having many (or even few) 'viable' autocatalytic cores in a GARD system is rather slim. The robustness of GARD cores to parasitic periphery is pinpointed. As a conclusion, it is suggested that a 'Lipid world' based on a simple artificial chemistry may harbor many cores, possibly infinite, facilitating evolutionary process.

Introduction

The 'Lipid world' scenario [Segre et al. 2001a] suggests that life may have emerged from non-covalent molecular assemblies. Such aggregates may be micelles, bilayers or vesicles that spontaneously rise when amphiphilic molecules (lipids) are introduced to an aqueous environment. Chemical studies and analysis of the scenario go back many decades [Oparin, 1957]. Theoretical studies of this scenario utilize the Graded Autocatalysis Replication Domain (GARD) model [Segre et al. 2000], which is briefly described in the following section.

Based on GARD simulations, it was postulated that self-sustaining autocatalytic networks lack evolvability [Vasas et al. 2010]. Farther studies on other models of autocatalytic networks [Kauffman, 1993] suggested that while most models fail to portray evolutionary capacity altogether, some may have such capacity, though more research is required in that realm [Vasas et al. 2012]. A detailed study, by the group that originated the GARD model, suggests that mutual catalysis is fundamental for evolvability [Markovitch and Lancet, 2012]. However, this later study (cf. figure 6 therein) show that a typical GARD system have a single compotype (conceptually a compotype may be thought of as an attractor or a quasistationary state in the network dynamics), and no simulation (out of 10,000) showed more than six compotypes.

Modeling the 'Lipid World'

The basic GARD model (dealt with hereafter) is thoroughly described in numerous publications (see ool.weizmann.ac.il for a partial list including several other variants of the model). Follows, a concise description of the model (adopted from [Vasas et al. 2010]):

It (the model – the author) involves discrete stochastic changes in noncovalent assemblies dictated by the differential equations N_G is the molecular repertoire of environmentally available

$$\frac{dn_i}{dt} = (\rho_i k_i N - k_{-i} n_i) \left(1 + \frac{1}{N} \sum_{j=1}^{j=N_G} \beta_{ij} n_j \right) i = 1, 2, \dots, N_G$$

prebiotic compounds; ρ_i is the external concentration of molecular species i; $k_i = 10^{-2} sec^{-1}$ and $k_{-i} = 10^{-5} sec^{-1}$ are uncatalyzed forward and backward rate constants assumed to be equal for all molecules for simplicity [they differ in their mutual rate enhancement properties]; $N(N < N_G)$ is the assembly size given by $N = \Sigma n_i$, with n_i indicating the count of molecular species i (within the assembly – the author) and β_{ij} is an element of the $N_G \times N_G$ positive matrix that defines the network of mutually catalytic interactions governed by a statistical formalism (see below)

The course of the composition of an assembly is governed by the equations above. It was shown that if assembly expansion is not disrupted, the assembly reaches an asymptotic steady-state composition [Segre et al. 2001b]. A more complex dynamics rise when a GARD assembly goes through a growth-fission process. Fission of molecular assemblies is likely to occur as the assembly grows larger. In GARD this was simplified proposing that as the assembly size (N) doubles the assembly breaks into two daughter assemblies, where each molecule in the parent assembly has 50% probability to go to either daughter assembly. Adding fission process keeps the assembly out of equilibrium. Fission is necessary (though not sufficient) to allow the exsistence of several quasistationary compositions (composomes) in a GARD system.

The matrix β defines the amount of mutual catalysis exerted on the join/leave reactions depicted by the kinetic equations. The elements β_{ij} of the matrix are drawn from a log-normal distribution with parameters μ =-4 and σ =4. The log-normal distribution has a long tail, allowing seldom high values, often many orders of magnitude above the background average. The log-normal distribution is an approximation of the Receptor Affinity Distribution [Lancet, 1993] modified for catalytic rate enhancement.

Results

The values in the β matrix are generated independently. Thus, the distribution of values in the j^{th} row of β (that is the catalysis exerted by molecular species j on all N_G molecular species) is similar to the distribution of the values in whole β . The probability that the largest value in the j^{th} row will occur in the j^{th} column (i.e. on the diagonal of β) is $1/N_G$. The probability that for all N_G rows the largest value (of each row) do not occur on the diagonal is therefore:

$$(1-\frac{1}{N_{\rm G}})^{N_{\rm G}}$$

which is approximately 1/e or 36.79%. Hence, most GARD systems (63.21%) are likely to hold at least one auto-catalytic molecular species. We shall denote these systems as Class-1.

Now, let us consider other classes of systems. Class-2 systems harbor a core (cf. Vasas et al. 2012) of two molecular species (if molecular species A and B are the core, the strongest catalysis of A is on the reaction that generates B and the strongest catalysis of B is on the reaction that generates A), but do not hold an auto-catalyst. Class-3 systems are neither in Class-1 nor in Class-2. Similar to analysis in the previous paragraph, the probability for a system to be in Class-3 is:

$$\left(1 - \frac{1}{N_{\rm G}} - \frac{N_{\rm G} - 1}{N_{\rm G}} \frac{1}{N_{\rm G}}\right)^{N_{\rm G}}$$

which is approximately 1/e squared or 13.53%. This leads to the conclusion that the probability of a system to be in Class-2 is 23.25%. (36.79%-13.53%).

It should be noted that given the characteristics of the distribution that generates the entries of β the largest value in a row is very likely to be larger than the sum of all other entries in the row. Consequently, the effect of parasitic periphery on the sustainability of the core diminishes. This strength of the GARD model is generally ignored elsewhere, and should be further analyzed.

The detailed outcome of a specific GARD system is determined by the holistic contribution of the catalysis. Yet, the dynamic corresponds, usually, to the classification of the system:

- Class-1 systems are generally governed by the strongest auto-catalyst that out-competes all other auto-catalysts. The governing compotype may "vanish" for some periods where fabricated-composomes (a very strong catalyst that "pumps" its substrate on expense of other species as long as the catalyst is within the assembly). Conceivably, other cores, if exist, may compete with the governing autocatalyst. Yet, as larger cores lack robustness (compared to the auto-catalyst) they rarely prevail.
- Class-2 systems show, typically, a mixture of core based composomes (if several small cores exist in the network) and fabricated-composomes as well as some periods of random compositional drift.
- Class-3 systems are generally in a drift state as with seldom appearances of fabricated-composomes. The large core generally cannot "squeeze" into the assembly, in line of the analysis briefly portrayed earlier [Shenhav et al. 2004]

The analysis and study of GARD classes is in progress.

Discussion

The theoretical analysis above is in line with the empirical results that GARD systems commonly show a single or at most few compotypes. Nonetheless, based on the analysis, though requiring additional effort, GARD systems may be classified according to features in their underlying network (which may be a rough analogy to genotype) rather than according to the dynamics the system manifests (phenotype).

GARD was severely criticized for its lack of evovability, Here, it is suggested that this deficiency is due to the nature of the underlying distribution of catalysis commonly used in the model. It is likely that engaging with an alternative structure for the β matrix, for example a block-diagonal, should result to multiple 'viable' cores, with possible better evolvable capacity. Amending the underlying chemistry resembles suggestions by others [e.g. Giri and Jain, 2012] regarding "fixing" a delinquent model. While the chemical plausibility of such matrix structure may be questioned, pursuing this kind of artificial chemistry seems promising.

The extension of the analysis to other GARD variants [Shenhav et al. 2007] is far from straight-forward.

References

- Giri V. and Jain (2012). The origin of large molecules in primordial autocatalytic reaction networks. PLoS ONE, 7(1): e29546
- Kauffman S. A. (1993). Origins of Order. Oxford University Press, New York, NY.
- Lancet D., Sadovsky E. and Seidemann E. (1993). Probability model for molecular recognition in biological receptor repertoires -Significance to the olfactory system. Proceedings of the National Academy of Sciences of the United States of America, 90(8):3715– 3719
- Markovitch O. and Lancet D. (2012). Excess mutual catalysis is required for effective evolvability. *Artificial Life*, 18(3):243–266.
- Oparin, A. I. (1957). The Origin of Life on the Earth. Oliver and Boyd, London
- Segre D., Ben-Eli D. and Lancet D. (2000). Compositional genomes: prebiotic information transfer in mutually catalytic noncovalent assemblies. Proceedings of the National Academy of Sciences of the United States of America, 97(8):4112–4117.
- Segre D., Ben-Eli D., Deamer D. and Lancet D. (2001a). The lipid world. Origins of life and evolution of the biosphere, 31:119–145.
- Segre D., Shenhav B., Kafri R. and Lancet D. (2001b). The molecular roots of compositional inheritance. *Journal of Theoretical Biology*, 213:481–491.
- Shenhav B., Kafri R. and Lancet D. (2004). Graded artificial chemistry in restricted boundaries. In Pollack J., Bedau M.A., Husbands P., Watson R.A. and Ikegami T., editors, *Artificial Life IX*, pages 501– 506. MIT Press, Cambridge, MA.
- Shenhav B., Oz A. and Lancet D. (2007). Coevolution of compositional protocells and their environment. *Philosophical Transactions of the Royal Society B*, 362: 1813–1819.
- Vasas V., Szathmary E. and Santos M. (2010). Lack of evolvability in self-sustaining autocatalytic networks constraints metabolism-first scenarios for the origin of life. *Proceedings of the National Academy of Sciences*, 107(4):1470-1475.
- Vasas V., Fernando C., Santos M., Kauffman S. and Szathmary E. (2012). Evolution before genes. *Biology Direct*,7:1

Spatial Patterning with the Rule of Normal Neighbors

Micah Brodsky¹

¹MIT Computer Science and Artificial Intelligence Laboratory micahbro@csail.mit.edu

Abstract

Based on developmental biology's Rule of Normal Neighbors, we develop a new mechanism for spatial patterning, exhibiting spontaneous symmetry breaking, regeneration, and approximate scale invariance. The desired pattern is represented as a topological adjacency graph, yielding an energy function that cells minimize through local interactions. Combined with a controller manipulating cells' mechanical properties, we demonstrate programmable geometric homeostasis for 3D cellular surfaces via simultaneous patterning and deformation.

Introduction

How might one assemble a pattern from un-differentiated tissue or regenerate a missing piece lost to injury? A common theme seen in developmental biology is the Rule of Normal Neighbors (Mittenthal, 1981): a point in a patterned tissue knows what elements of the pattern belong adjacent to it, its "normal neighbors". If it finds its neighbors are wrong, it takes steps to correct the situation, such as re-growing a more appropriate neighbor or changing its own fate to better fit its environment. This general rule captures many striking experimental results, such as the growth of inverted segments in cockroach limbs when the distal portions of the limbs are excised and replaced with longer explants (French, 1981).

In recent work (Brodsky, 2014a), we propose one possible mechanism by which patterning and pattern repair under the rule of normal neighbors can be implemented, with only local computation and minimal resources. We represent the topology of a desired pattern as an adjacency graph over discrete pattern states. Based on this graph, we construct an energy function using local interactions for which the desired pattern is (usually) a minimum. Cells can then explore this potential by a process mathematically analogous to thermal (and simulated) annealing, seeking a minimum.

The result is a spatial patterning mechanism that demonstrates spontaneous symmetry breaking, regeneration, and approximate scale invariance. In conjunction with control algorithms for cell-cell traction and bending forces, it has been used to generate self-stabilizing 3D geometries within a simplified model of embryonic tissue. This serves a primitive artificial demonstration of morphological homeostasis.

Energy specification

The core idea of the rule of normal neighbors is purely topological: what can lie adjacent to what. This alone is not enough to form a pattern, but it's a good starting place. In our

formalism, each region of the pattern is assigned a discrete state, and the adjacency graph captures the neighbor relationships between homogeneous regions of a single state (see Figure 1). An implicit self-edge exists for each state, in order that the representation be scale-invariant and meaningful both in continuum and on a discrete lattice.

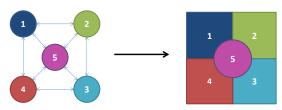


Figure 1 – Example adjacency graph (left), associated desired pattern (right).

A variety of other patterns will satisfy the same adjacency graph, however, indeed, any arbitrary continuous distortion of the original pattern. Furthermore, any simply connected (but possibly overlapping) cut through the pattern, such as a spiral that repeatedly cut through the same regions, will still satisfy as well, albeit with some regions and neighbor contacts duplicated and others absent.

In order to avoid arbitrarily pathological deformations, we favor compact, blob-like regions by including a virtual "surface tension" self-affinity term in the energy function. To avoid arbitrary cuts with missing regions, we can either impose boundary conditions that force every region contacting the boundary to appear in the pattern, or we can add perregion quorum sensing. The former will ensure that all boundary-contacting regions are present, although the sizes of non-boundary-associated regions will be unstable, and they may shrink to a sliver or disappear entirely. Alternatively, quorum sensing adds complexity but can ensure the stability of region sizes and, in conjunction with surface tension, strongly discourage the presence of duplicate regions.

Energy minimization

For suitably-constructed energy functions such as sketched above, gradient descent can be implemented purely locally, with local information and local updates. However, spatial patterns are prone to local minima, and gradient descent fails almost immediately. Instead, a probabilistic strategy is quite successful. Energetically favorable transitions are made with high probability and unfavorable transitions are made with low probability. With appropriate choice of weights (i.e. the

Boltzmann distribution), this becomes analogous to natural and simulated annealing, complete with a "temperature" parameter.

The algorithm sketched thus far works fairly well. However, it is noisy and has no clear termination condition. As an alternative, we can formulate a "thermodynamic limit" to the stochastic algorithm, where fluctuating discrete states are replaced by mean field values. The result is somewhat analogous to loopy belief propagation and produces clean, robust patterns even at significant temperatures, suitable for driving downstream actuators.

Regulating Geometry

Ultimately, the goal of patterning is to spatially choreograph phenotypic properties. These properties may be simple and self-contained (e.g. "color"), or they may be disruptive and far-reaching, such as the geometry of the substrate itself. With substrate manipulations, the patterning process is not a feed-forward cascade but is in general a large feedback loop, changes to geometry rearranging and disrupting the original pattern. However, given a suitably robust and self-stabilizing patterning mechanism such as above, combined with suitable closed-loop controllers for geometric features, self-stabilizing geometry that is faithful to the pattern can be demonstrated. With a simple embryonic epithelium model (Brodsky, 2014b), actuation mechanisms that combine curvature sensing, intrinsic forcing (e.g. purse-string), and extrinsic bending (apical/basal constriction) have been shown to be successful.

Results

Several successful patterns and their associated adjacency graphs are illustrated in Figure 2, using the mean field

algorithm. These patterns can self-organize on a variety of domain shapes and sizes and can self-repair in response to large sections being erased. Temperature was selected or annealed as appropriate for the size of the domain and the complexity of the pattern; under poor temperature conditions or with certain pathological patterns, topological defects can arise, particularly twinning (duplication of regions and subpatterns) and twisting (irreconcilable partial mirror inversions). The rightmost example illustrates a case of the algorithm operating on and directing a dynamically deforming surface (Brodsky, 2014b), such that the overall shape is determined by the underlying pattern. Under the direction of the algorithm, cells flow and rearrange themselves to produce the desired structure. Such self-organizing, self-stabilizing geometries are the subject of ongoing work.

Acknowledgments

The author would like acknowledge Gerald J. Sussman for supervising this work. This material is based on work supported in part by the National Science Foundation under Grant No. CNS-1116294 and in part by a grant from Google.

References

Brodsky, M. Z. (2014a) Patterning with the Rule of Normal Neighbors. MIT CSAIL tech. rep., in preparation.

Brodsky, M. Z. (2014b) Deformable Amorphous Computing with Foam-Inspired Surface Mechanics. MIT CSAIL tech. rep., in preparation.
French, V. (1981) Pattern regulation and regeneration. *Phil. Trans. Royal Soc. London. Series B, Biological Sciences*, Vol. 295, No. 1078.

Mittenthal, J. E. (1981) The rule of normal neighbors: A hypothesis for morphogenetic pattern regulation. *Developmental Biology*, 88(1):15-26.

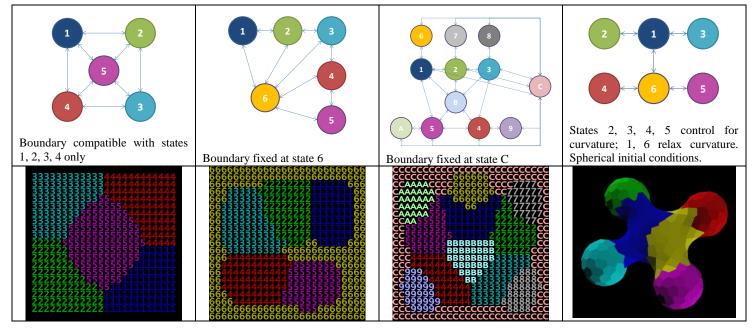


Figure 2 – Example successful results – exhibiting exactly one instance of every region, each contacting every normal neighbor and no spurious neighbors – achieved under appropriate annealing conditions. Left three are on rectangular lattices, while rightmost is on an irregular, dynamically changing cellular mesh of spherical overall topology.

CodeRouge: a Project to Evolve Life-like Autonomous Programs

Nawwaf N. Kharma¹ and William R. Buckley²

¹ECE Dept., Concordia University, 1455 de Maisonneuve Blvd. O., Montreal, QC, Canada H3G1M8

²California Evolution Institute, San Francisco, CA, USA 94134

¹kharma@ece.concordia.ca and ²wrb@calevinst.org

Abstract

The aim of this project is to create a computational environment that allows for the design/evolution of programs with life-like behavior. By life-like behavior we mean programs whose main aim is to exist and reproduce within their environment, and exhibit other essential signs of life: homeostasis & adaptation, growth & open-ended evolution. In order to give digital organisms a functionality of use to humans, a program will also be able to carry out, and autonomously improve upon, a human defined activity. For many reasons, we have chosen to build a computational environment (in emulation) and a new redcode-like language, µRouge, which runs in it. This paper describes the concepts and instructions of this new language and provides an example highlighting some of its unusual characteristics.

Motivation

The existing redcode of Core War (Dewdney, 1984; Jones and Dewdney, 1984) allows for the development of novel algorithms through the direct manipulation of program instructions but, it does not allow for those novel algorithms to be shared among a population of programs. The current effort extends redcode to provide explicit facilities for such algorithm sharing, and so increasing evolutionary pressure on shared algorithms. Further, evolution in Core War redcode programs is ungoverned save solely the purpose of execution longevity. We propose an added level of selection within the model, by means of an auxiliary task; satisfaction of this task concomitant with steadily improving life scores is critical to the continued computation of the program. Ultimately, we wish to build a programming language and platform that allow for the evolution of programs with life-like characteristics- characteristics that are discussed below.

Existence

A program exists if its body is present in, or can deploy to, working memory (~ space), where it can access processing time (~ energy).

Metabolism

A program has a metabolism if it is using processing time to execute instructions. The environment is any space with a state that can be altered, computationally, which includes working memory as well as long-term memory.

Growth

Growth is an increase in the size of a program. The size of a program is the amount of working memory it occupies, when active (i.e., has a metabolism).

Reproduction

Reproduction of a program is the synthesis of a copy of the program identical/similar to all/part of the parent, a copy that occupies memory space outside the boundaries of the memory space of the parent. Copying occurs imperfectly.

Darwinian Reproduction. In this mode of reproduction, a program copies a compressed self-description of its *original* body to a free location in memory, also copies a compression/decompression routine (or *comdec*) to that location, then applies it to the compressed self-description, to complete the synthesis of the body of the child.

Lamarckian Reproduction. In this mode of reproduction, a program first creates a compressed self-description of its *current* body, then it copies that together with the comdec to a free location in memory. Hence, the program applies the comdec to the compressed self-description in order complete the synthesis of the body of the child.

Homeostasis

Homeostasis of a program is its ability to maintain or regain its functionality (possibly imperfectly) in response to a perturbation to its body or environment (e.g., input). To achieve this, we assert that (a) Minimal changes (or *mutations*) to either input data or instructions will have no impact on the meaning of that data or instructions; (b) greater (non-minimal) mutations to an instruction or datum can only result in proportional changes in the meaning of that instruction or datum; (c) no amount of mutation will render an instruction or datum meaningless; (d) instructions can be interpreted as data and data as instructions.

Adaptation

A program adapts by changing its body or that of its descendants in response to changes it senses in its body or environment. A program can spring-off mutated copies of itself, but it can also modify its own body during its existence.

Evolution

Both flavors of reproduction result in a diverse populationselection is needed. Selection can be explicit (and purposeful: see Utility) or implicit. Implicit selection occurs if, for example, smaller programs are allowed to replicate faster than larger programs. It is important that we do not unconsciously implicitly favor one kind of program over another. Also, to allow for open-ended evolution, our model permits the runtime definition of new instructions.

Utility

A particular part of a program (called human load) is assessed for its effectiveness in fulfilling a human-specified functionality, and for efficiency, represented by size.

Program Structure & Instruction Set

A program is made of a circular linked list of program lines. A program line has the format: <descriptor> <instruction-modifier><data source & destination><next instruction pointer><error detection code>. An instruction is a traditional instruction (INS), a new instruction (NIN), a data line (DTA) or a free line (FRE). An INS indicates a pre-defined instruction that transforms data (e.g., ADD). A NIN says that this instruction was defined by the program itself at run-time. DTA indicates data. FRE says that this space is free to be used as local working memory.

Instruction Set

Arithmetic & Logical Operations. Arithmetic addition (ADD), subtraction (SUB), multiplication (MUL) and division (DIV) are all implemented. Logical AND, OR (ORR), NOT and XOR are implemented. Shift left (SHL) and shift right (SHR) are also implemented.

Program Flow Control. An unconditional jump (JMP) and conditional jumps on zero (JMZ) and not-zero (JMN) are implemented. Also, jumps conditional on the equality (CMP) or inequality (SLT) of the first two operands are implemented. Finally, we implemented a jump (DJN) that first decrements the second operand then jumps to a location specified by the first operand only if the result of the decrement is not zero.

Self-Sensing Instructions. A critical subset of instructions are those providing a program with information about itself. These include: age (AGE), which returns the number of cycles that passed since the program was first instantiated; size (SIZ), which returns the number of lines a program is made of; metabolism (MET), which is a measure of program activity- a function of the number of executed instructions; homeostasis (HOM), which is a measure of program health: freedom from errors; growth (GRW), which is a ratio of the program's current size to its initial size; offspring (OFS), which is the number of offspring a program created since its instantiation; number of new instructions (NNI), which is the number of new instructions (automatically defined functions, in fact) that were defined by the program since its instantiation; fitness (FIT), which is a measure of the performance of the human load.

Self-Modification Instructions. Another critical set of instructions are those that allow a program to alter its own 820

code. Edit (EDT) allows for the modification of a line (or part thereof), at a given location within the program, in a particular manner. Generate random (GRN) inserts a randomly generated instruction or data line at a given location within the program. The copy sequence (COS) instruction and move sequence (MOS) instruction, respectively, copies or moves a number of contiguous program lines to a certain location within the program. Copying can occur w/without mutation, w/without compression/decompression, and by overwriting or down-shifting the lines at the insertion point.

Program Reproduction. There are two instructions that implement the two modes of replication described above: REP, which realizes Darwinian reproduction and CON, which implements Lamarckian reproduction.

New Instruction Definition. This allows for the run-time definition of new instructions, in an analogous fashion to automatic function definition (Koza, 1992).

Miscellaneous Instructions. A few other instructions (5) are added for pragmatic considerations.

Program Example

	MET	\$F,	\$G,	#0	;metabolism
	OFS	\$C,	#0,	#0	;offspring
	NI1	\$C,	\$F,	\$S	;new inst. 1
	CMP	\$C,	\$S,	# O	;ready?
	JMP	\$0,	#O,	<t< td=""><td>;if not, exit</td></t<>	;if not, exit
	COS	\$SR,	#LN,	\$DS	;copy subrtn
	EDT-REP	\$ALT,	\$DS+n,	#7	;edit opr c
	SPL	\$DS+e,	<sct,< td=""><td>#0</td><td>;new thread</td></sct,<>	#0	;new thread
0:	ADD	#1,	\$PCT,	#0	;continue

The example shows the use of program health measures, metabolism and offspring, in controlling both the selection of other subprograms for replication, the spawning of a new thread of execution corresponding to such a subprogram replication event, and a post-replication change to the compression-decompression algorithm used for subsequent subprogram replication tasks.

Here, the health measures metabolism (MET) and offspring (OFS) are combined by a new instruction (NI1) to generate a testable condition, stored in location \$S. This condition then governs the spawning of a new process; if ready, a copy of the source code is placed at the destination address, and a targeted REP instruction of the copy is also modified, replacing the comdec. It is this modified program that is spawned by the SPL instruction.

References

Koza, J. R. (1992). Hierarchical automatic function definition in genetic programming. In Proceedings of Workshop on the Foundations of Genetic Algorithms and Classifier Systems.

Dewdney, A. K. (May 1984). Computer Recreations, Scientific American.

Jones, D. G., Dewdney, A. K. (March 1984). Core War Guidelines. http://corewar.co.uk/cwg.txt.

The Effect of Connection Cost on Modularity in Evolved Neural Networks

Jessica Lowell¹ and Jordan Pollack¹

¹DEMO Lab, School of Computer Science, Brandeis University, Waltham, MA 02453 jessiehl@cs.brandeis.edu

Abstract

Modularity, often observed in biological systems, does not easily arise in computational evolution. We explore the effect of adding a small fitness cost for each connection between neurons on the modularity of neural networks produced by the NEAT neuroevolution algorithm. We find that this connection cost does not increase the modularity of the best network produced by each run of the algorithm, but that it does lead to increased consistency in the level of modularity produced by the algorithm.

Introduction

As evolutionary systems are increasingly used in a variety of applications, the organization of the evolved solutions has become important. Modularity, the organization of a system into interacting subparts, is observed in both many natural and many engineered networks (Koza, 1992; Simon, 1996; Hartwell et al., 1999; Wagner and Altenberg, 1996). We briefly discuss modularity in natural and simulated evolution, as well as the NEAT neuroevolution algorithm, a widely used evolutionary algorithm.

Modularity

Both biological networks, such as neural networks and bacterial metabolic networks, and biological systems in general, such as tissues assembled from cells, tend to be modular. The evolutionary reasons for this are still unclear, especially because computational models of biological evolution tend to produce nonmodular solutions. The nonmodular solutions produced by network-evolving algorithms are often connected in complicated ways and better-performing on the specific task for which they are optimized than modular solutions designed by humans (Thompson, 1998; Vassilev et al., 2000). However, this lack of modularity in computational evolution means that while it can produce highly optimized solutions for simple problems, it has difficulty solving more complex problems (Kashtan and Alon, 2005).

Some work has been done on creating modularity through evolutionary algorithms by building the encapsulation of modules into the algorithms (Garibay et al., 2004; Wiegand et al., 2009). This is sufficient if the goal is simply to create modular solutions. However, if the goal is to understand how modularity evolves in nature, imposing modularity on the algorithm does not suffice. Furthermore, even if the goal is an engineering one rather than one of biological discovery, allowing the evolutionary process to discover modularity naturally ensures that high-performing species of solutions are not excluded from the search space from the start because they began as strongly nonmodular and took many generations to evolve modularity.

How modularity evolves in both natural and simulated evolution is a complex problem that may involve multiple forces whose contributions must be teased out. A major hypothesis is that it evolves in response to varying environments (called modularly varying goals), in which solutions must perform different tasks with common subtasks (Kashtan and Alon, 2005; Kashtan et al., 2007). This effect was proposed by Lipson et al. (2002) in a study of minimal substrate modularization which found that modular separation is logorithmically proportional to rates of environmental variation, and suggested that evolutionary design of engineered systems should use variable rather than fixed fitness critiera. The effect of modularly varying goals has been observed in both computational (Kashtan and Alon, 2005) and natural (Kashtan et al., 2007; Parter et al., 2007) evolution studies. An alternative hypothesis, supported by the work of Clune et al. (2013), is that modular networks evolve in response to a "connection cost", or small decrease in fitness for each connection in the network, analagous to the energy cost of forming a physical link between two cells, organisms, or other nodes in the biological network. The debate over which of these hypotheses, if either, plays the larger role in the evolution of modularity, is ongoing. We investigated the connection cost hypothesis as applied to NeuroEvolution of Augmenting Topologies (NEAT), a major method for evolving artificial neural networks. Future work may compare both hypotheses on NEAT or another neuroevolution algorithm.

NEAT

There are numerous algorithms for evolving neural networks, some of which evolve topology only or weights only, and some of which evolve both. NEAT (Stanley and Miikkulainen, 2002) is an example of a neuroevolution algorithm that evolves both topology and weights. It starts with very simple networks and gradually complexifies candidate solutions using crossover and three forms of mutation: adding neurons, adding connections, and changing connection weights. To protect innovations long enough to see if they will be evolutionarily useful, it also uses speciation, which isolates subsets of the population into reproductive groups based on how different they are topologically. Finally, it tracks the history of innovations through historical innovation numbers to mitigate the competing conventions problem. NEAT has proven useful in several problem domains (Stanley and Miikkulainen, 2002, 2004), and is one of the most popular neuroevolution algorithms.

The standard NEAT algorithm does not tend to produce modular solutions (Reisinger et al., 2004). Reisinger et al. (2004) created Modular NEAT, a version of NEAT that does produce modular solutions, but forced the algorithm's predisposition toward modularity by requiring it to reuse neural substructures in different spatial locations to form complete neural networks. Clune et al. (2010) found that Hyper-NEAT, a variant of NEAT, did not tend to produce modular networks. Verbancsics and Stanley (2011) were able to influence HyperNEAT toward modularity by seeding it with a bias toward local connection - the connection of components that are spatially near each other - which is another force, besides those previously mentioned, that may play a role in biological modularity.

In this paper, we take the connection cost hypothesis and apply it to NEAT in order to study whether it influences the emergence of modularity in networks produced by NEAT in the same way that it influenced the emergence of modularity in other problem domains in (Clune et al., 2013). This includes the level of modularity, but it also includes variance in modularity. In other words, we ask whether connection cost makes the amount of modularity emerging from NEAT more consistent from trial to trial, in addition to looking at its influence on the overall amount of modularity.

Methods

To test the effect of connection cost on modularity in NEAT, we used two different versions of the eight-pixel retina problem, which was developed by Kashtan and Alon (2005) for studies on the emergence of modularity, and which has been used in other studies on the emergence of modularity (Clune et al., 2010; Verbancsics and Stanley, 2011). In the retina problem, neural networks attempt to recognize certain patterns (objects) in an four-pixel by two-pixel retina, and return true or false based on whether those objects are present. The first is modularly decomposable, in that the neural net-

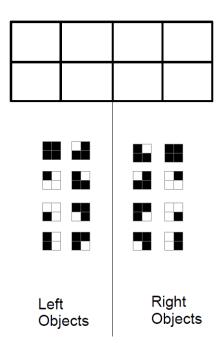


Figure 1: Illustration of left and right objects in the eightpixel retina problem (adapted from Kashtan and Alon (2005)).

work has to determine whether objects exist on each of the left and right sides (which each contain four pixels) in order to determine whether objects exist on both sides. The second version of the problem is not modularly decomposable, in that an object need only exist on one side in order for the function to return an output of true. Which patterns count as objects are slightly different for the left and right sides of the retina for both versions of the problem. This is illustrated in Fig. 1.

We ran the NEAT evolutionary process on both versions of the problem, using the NEAT4J open source Java implementation of NEAT (Simmerson, 2006) as a basis, with and without connection cost. The parameters used are listed in Table 1.

Problem Ve	rsion	Trials	Connection Cost
Modula	ır	20	0
Modula	ır	20	1.0×10^{-5}
Nonmodu	ılar	20	0
Nonmodu	ılar	20	5.0×10^{-5}

Table 1: Retina problem versions and parameters used.

The connection cost for each problem was determined by testing several different orders of magnitude for connection cost, and then several different connection costs in intervals of 0.00001, looking for the highest cost that consistently allowed solutions to evolve performance improvements of 10% or greater over the first-generation solution, a fully-connected network of eight input and one output neurons with no hidden layer.

Each run of the NEAT algorithm used a population of 500 neural networks evolving over 2000 generations. Fitness was measured by mean standard error on the problem, meaning that a lower fitness number indicated better performance.

We quantified modularity by using the metric Q, defined by the approach of Newman and Girvan (2004). This approach determines Q by looking at the percentage of edges in the network that connect nodes in the same module, and substracts the expected value for that percentage in a network with the same number of modules but random connections. The modules are defined by a previous part of the algorithm that splits the network into the modules that would maximize Q. Mathematically, the equation for Q is:

$$Q = \sum_{s=1}^{k} \left[\frac{l_s}{L} - \left(\frac{dS}{2L} \right)^2 \right] \tag{1}$$

where L is the number of edges, K is the number of modules, d_s is the sum of degrees of nodes in module s, and l_s is the number of edges in that module.

To determine whether variances in modularity were equal in our sets of results, we used Levene's test, a statistic for assessing the equality of variances across two or more groups. We used Levene's test rather than an F-test of equality of variances because the F-test is highly sensitive to non-normality of distribution, while Levene's test is robust to non-normality.

Results and Discussion

Our first comparisons were between a set of 20 trials without a connection cost and 20 trials with a connection cost of 1.0×10^{-4} per connection, with mutation probability = 0.25 and crossover probability = 0.2. In Fig.2, we can see that the best solutions produced in the trials with no connection cost had a variance in modularity of 0.0449, while those produced in the trials with a connection cost had a variance of only 0.0177 - more than 60% lower. This difference in variance was statistically significant (P = 0.001803).

We examined the possibility that the difference in variances was caused by interactions between crossover and connection cost, rather than connection cost alone. When we removed crossover but kept all other parameters the same, we obtain similar results, as seen in Fig.3. In this case, the variance in modularity for the trials with no connection cost is 0.0455, while the variance with connection cost is 0.0182.

When we underwent the same process using the nonmodular version of the retina problem, we still saw decreases

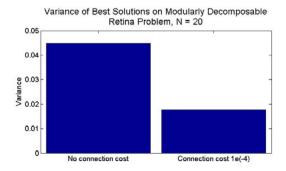


Figure 2: The addition of a connection cost to NEAT, with both mutation and crossover, on the modularly decomposable retina problem produces a decrease in variance across trials. Number of trials N = 20, P = 0.001803

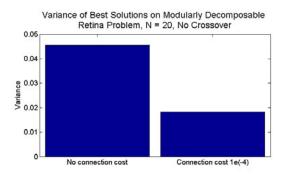


Figure 3: The addition of a connection cost to NEAT, with mutation but without crossover, on the modularly decomposable retina problem produces a decrease in variance across trials. Number of trials N = 20, P = 0.006627

in variance, but they were just below the level of statistical significance, in the P = 0.06-0.07 range.

With both mutation and crossover, the variance without connection cost was 0.0189 and the variance with a connection cost of 5.0×10^{-4} per connection was 0.0098 (Fig.4). With mutation but no crossover, the variance without connection cost was 0.0189 and the variance with a connection cost of 5.0×10^{-4} per connection was 0.0078 (Fig.5).

We also examined the effect of crossover itself on modularity, in order to further separate any of its effects from those of connection cost. There was no significant difference between the variances of any set of trials with crossover and the otherwise-equivalent set without crossover (P = 0.617905).

This reduction in variance caused by connection cost represents an increase in predictability. In the presence of connection cost, different runs of NEAT are more likely to produce similarly modular solutions. This suggests that in situations where connecting between nodes involves a physical link and thus an energy cost, there may be an optimal

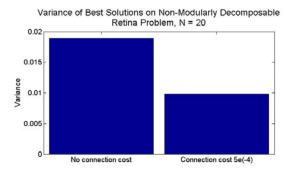


Figure 4: The addition of a connection cost to NEAT, with mutation and crossover, on the non-modularly decomposable retina problem produces a sub-statistically-significant decrease in variance across trials. Number of trials N=20, P=0.061516

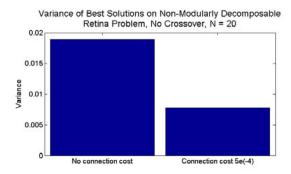


Figure 5: The addition of a connection cost to NEAT, with mutation but without crossover, on the non-modularly decomposable retina problem produces a sub-statistically-significant decrease in variance across trials. Number of trials $N=20,\,P=0.062036$

level of modularity, a balance between modularity and efficient use of space. Such a balance may be related to work on wiring economy in the human brain described by Bullmore and Sporns (2012), which found that the brain balances modularity with efficient use of space and that an imbalance in either direction causes neurological disorders.

We include images of a few sample networks from our experiments, with varying structures and amounts of modularity. Fig. 6 is a network with Q=0 (all nodes are found to be part of a single module by the Newman-Girvan algorithm), produced by running NEAT4J on the modularly decomposable test problem with no connection cost. Fig. 7 was also produced by running NEAT4J on the modularly decomposable test problem with no connection cost, and is a large network with a high modularity of Q=0.4567, divided by the Newman-Girvan algorithm into 13 modules. Fig. 8 is a small network with a moderate modularity of Q=0.1760, produced on the nonmodularly decomposable test problem

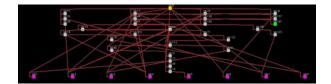


Figure 6: Example network created by running NEAT4J on modularly decomposable test problem with crossover. Q = 0, 37 neurons.

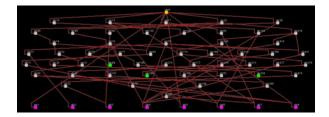


Figure 7: Example network created by running NEAT4J on modularly decomposable test problem with crossover. Q = 0.4567, 61 neurons, 13 modules.

with a connection cost, divisible into 6 modules. In these images, input neurons are pink, output neurons are orange, neurons that are not connected to any input neuron are green, and all other neurons are gray.

An observer can see from these networks that modularity is not necessarily a function of network size. However, connection cost did appear to promote smaller networks - in experiments with crossover, average network sizes on the modularly decomposable problem were 47.1 neurons with connection cost vs 62.75 neurons without, and average network sizes on the nonmodularly decomposable problem were 30.6 neurons with connection cost vs 38.4 without. It is possible that the connection cost is preventing new nodes and links from being formed, which we might expect since it slightly penalizes connections between nodes, without necessarily increasing modularity in the process. One reason that connection cost may promote the evolution of modularity in some cases is that in a modular system there are fewer connections between nodes. When nodes and links evolve together, fewer links may also encourage fewer nodes.

It is worth considering whether connection cost had an effect on fitness itself, since if NEAT could maintain fitness in the presence of connection cost, this would present a problem for using connection cost for engineering purposes. As we previously stated, fitness was represented by mean standard error, with a lower fitness being better. On the nonmodularly decomposable problem, fitnesses ranged from 0.0777 to 0.3807, while on the modularly decomposable problem, they ranged from 0.0408 to 0.2856. Connection cost made no statistically significant difference to either the variances of fitnesses or the mean fitness on either problem. Fitness

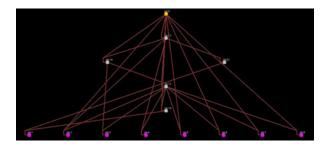


Figure 8: Example network created by running NEAT4J on modularly decomposable test problem with crossover. Q = 0.1760, 16 neurons, 6 modules.

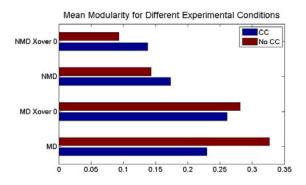


Figure 9: Mean modularity across 20 trials for all experimental conditions, with direct comparisons in the same color. MD represents the modularly decomposable retina problem version, NMD represents the nonmodular, "Xover 0" indicates that crossover was not used as an evolutionary operator. No direct comparisons showed statistically significant differences. Comparisons of sets with statistically equal variances were done using an unpaired t-test, others were done using Welch's t-test.

was maintained, though not improved, in the presence of connection cost.

There was no statistically significant effect of connection cost on the *magnitude* of modularity for either version of the retina problem (Fig.9). This may seem rather surprising given the apparent contradiction with the results of Clune et al. (2013), discussed earlier. This raises the possibility that some aspect of the NEAT algorithm itself prevents connection cost from increasing modularity.

A possible explanation for this difference with previous work on connection cost and modularity is NEAT's built-in protections against bloat, which do not exist in many other evolutionary algorithms. NEAT complexifies its structures slowly, and separates sufficiently different topologies into species so that they do not interfere with each other. The speciation also means that as long as simpler networks are competitive, they will survive in the population, as their suffi-

ciently complex offshoots will branch into different species. Connection cost and bloat reduction both make it less beneficial for the evolutionary algorithm to form the large, intricately-wired, highly-nonmodular networks that are often the result of evolutionary algorithms, and so the effect of connection cost on the modularity of solutions produced by NEAT may be somewhat redundant.

While none of the results in Fig. 9 were statistically significant, it is interesting to note that for the nonmodularly-decomposable problem there was a trend toward higher modularity with connection cost, but that with the modularly-decomposable problem the trend ran in the opposite direction, with connection cost correlating with lower modularity. The latter trend may be a side effect of the lowering of variance - in a relatively small population, there were fewer high-modularity outliers to bias the mean upward. However, it may also suggest that modularity emerges differently on modularly decomposable and nonmodularly-decomposable problems.

As we previously mentioned, Verbancsics and Stanley (2011) were able to increase the modularity of networks produced by HyperNEAT, a generative, hypercube-based, extension of NEAT, by starting the algorithm with a bias toward local connectivity. Local connectivity is also a part of the basis for the hypothesis that connection cost increases modularity in general. The fact that this failed to happen in our experiments with NEAT raises the question of whether local connectivity has different effects on evolution of modularity depending on whether the network being evolved has a generative or indirect representation, as is the case with HyperNEAT, or a direct representation, as is the case with NEAT. One possible reason for this is that in generative and indirect systems, the evolution of modularity-promoting factors and performance-promoting factors can be separated - for example, in the aforementioned extension of Hyper-NEAT, weights and connection expression patterns can be evolved separately.

In these experiments, we used a form of connection cost that assumes a constant cost per connection, rather than one that assumes a greater cost for a longer connection (for instance, one based on the principle that in a physical brain there would be a greater energy cost in creating an axon and synapse between two neurons that are far apart than in creating them between two nearby neurons). We made this choice because Clune et al. (2013), using both forms of connection cost, found that there was little difference between them in promoting modularity, and because NEAT does not have a built-in concept of physical distance between neurons. It may be that if an extension of NEAT that did have this concept built-in was developed, different forms of connection cost would have different effects on the evolution of modularity in NEAT-produced networks. In order to obtain the modularity-promoting benefits of indirect representations, as was done with HyperNEAT, the geometry of the network would need to be evolved separately from other aspects of the network.

These results suggest multiple possible directions for future work. We have already mentioned the possibility of developing a version of NEAT that contains a concept of physical distance between neurons and studying how connection cost that is weighted by distance affects the evolution of modularity compared to simple connection cost per link. Another possibility is to investigate whether connection cost leads to more predictability in modularity in general, rather than just when NEAT is the evolutionary algorithm used. Still another is to study the effect of connection cost on populations of neural networks, rather than just the best network produced by each run of the algorithm. Since, as we have discussed, there may be properties particular to NEAT that influenced our results, it could be clarifying to see if the results are similar if a different neuroevolution algorithm is used. Finally, it may be worth testing further how modularity is influenced differently when evolution is occurring on modularly vs nonmodularly decomposable problems.

In the results reported here, we find that adding a connection cost to the NEAT algorithm does not significantly affect the modularity of the top networks produced by NEAT. We speculate that this lack of effect is caused by NEAT's protections against bloat. We find, however, that it does reduce the variance in that modularity, leading to a more consistent level of modularity in the resulting networks, and thus more predictability in the outcome of the algorithm.

Acknowledgements

This work was supported by NSF grant No. 1068620. Thanks to Kyle Harrington for his valuable feedback on the ideas in this paper.

References

- Bullmore, E. and Sporns, O. (2012). The economy of brain network organization. *Nature Reviews Neuroscience*, 13(5):336–349.
- Clune, J., Beckmann, B., McKinley, P., and Ofria, C. (2010). Investigating whether hyperneat produces modular neural networks. In *Proceedings of the 12th annual conference on Genetic and Evolutionary Computation*, pages 635–642. ACM.
- Clune, J., Mouret, J., and Lipson, H. (2013). The evolutionary origins of modularity. *Proceedings of the Royal Society B*, 280(1755):20122863.
- Garibay, I. I., Garibay, O. O., and Wu, A. S. (2004). Effects of module encapsulation in repetitively modular genotypes on the search space. In *Proceedings of the 6th annual conference on Genetic and Evolutionary Computation*, pages 1125– 1137. Springer.
- Hartwell, L., Hopfield, J., Leibler, S., and Murray, A. (1999). From molecular to modular cell biology. *Nature*, 402:C47–52.

- Kashtan, N. and Alon, U. (2005). Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences*, 102(39):13773–13778.
- Kashtan, N., Noor, E., and Alon, U. (2007). Varying environments can speed up evolution. *Proceedings of the National Academy* of Sciences, 104:13711–13716.
- Koza, J. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA.
- Lipson, H., Pollack, J. B., and Suh, N. P. (2002). On the origin of modular variation. *Evolution*, 56(8):1549–1556.
- Newman, M. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113.
- Parter, M., Kashtan, N., and Alon, U. (2007). Environmental variability and modularity of bacterial metabolic networks. BMC Evolutionary Biology, 7(1):169.
- Reisinger, J., Stanley, K., and Miikkulainen, R. (2004). Evolving reusable neural modules. In *Proceedings of the 6th annual conference on Genetic and Evolutionary Computation*, pages 69–81. Springer Berlin Heidelberg.
- Simmerson, M. (2006). Neat4j homepage. online, 2006.
- Simon, H. (1996). *The Sciences of the Artificial*. MIT Press, Cambridge, MA.
- Stanley, K. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.
- Stanley, K. and Miikkulainen, R. (2004). Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21:63–100.
- Thompson, A. (1998). Hardware Evolution: Automatic Design of Electronic Circuits in Reconfigurable Hardware by Artificial Evolution. Springer, NY.
- Vassilev, V., Job, D., and Miller, J. (2000). Towards the automatic design of more efficient digital circuits. In *Proceedings of the* Second NASA/DoD Workshop on Evolvable Hardware, pages 151–160.
- Verbancsics, P. and Stanley, K. (2011). Constraining connectivity to encourage modularity in hyperneat. In *Proceedings of the 13th annual conference on Genetic and Evolutionary Computation*, pages 1483–1490. ACM.
- Wagner, G. and Altenberg, L. (1996). Complex adaptations and the evolution of evolvability. *Evolution*, 50(3):967–976.
- Wiegand, R., Anil, G., Garibay, I. I., Garibay, O. O., and Wu, A. S. (2009). On the performance effects of unbiased module encapsulation. In *Proceedings of the 11th Annual conference* on Genetic and evolutionary computation, pages 1729–1736. ACM.

The Resilience of a Swarm Ecosystem Under Environmental Variation

Jessica Lowell¹, Kyle Harrington¹, and Jordan Pollack¹

¹DEMO Lab, School of Computer Science, Brandeis University, Waltham, MA 02453 jessiehl@brandeis.edu

Abstract

Evolving swarms can be used both to solve real-world problems and to study biological and ecological phenomena. We simulated an evolving swarm of birds under three different types of climate-change-related environmental variation - a temperate environment becoming tropical, a temperate enviroment becoming a desert, and a tropical environment becoming a desert. We found that desertification increased expirations within the swarm and decreased population stability. The direction of the variation - tropicalification or desertification - had a greater impact on the dynamics of the swarm than the degree of variation when it came to these outcomes. The environmental variation also affected the genetics of the birds, with decreased food availability leading to collision avoidance genes being downplayed, and searching behavior for food being changed. High-intensity environmental variation led to less genetic stability post-change than lowerintensity environmental variation.

Swarming and flocking behavior is ubiquitous throughout all scales of biological and physical systems. Swarming simulations were first developed by Reynolds (1987) using his boids, simple agents that moved according to a set of basic rules. Much later swarming behavior simulation work has focused on agent-based modeling, which is centered around the modeling of populations of individuals with rules governing their behavior (Mach and Schweitzer, 2003). Agent-based modeling has been used to study such subjects as the dynamics of mountian pine beetle infestations of forests (Perez and Dragicevic, 2010) and the dynamics of how bacteria aggregate to form microfilms (Lardon et al., 2011). Recent advances in robotics have made it possible to experiment with large-scale physical swarms of robots (Rubenstein et al., 2012).

Swarm flying behavior is increasingly an area of interest, with a variety of algorithms and applications being developed. Karaboga (2005) used simulations of bee swarm flying to develop a numerical optimization method, and Su et al. (2009) modeled flocking behavior in the presence of a group leader. Optimization of heterogenous swarms of flying agents is challenging, but is potentially very useful in applications such as crop polination (Nagpal et al., 2011;

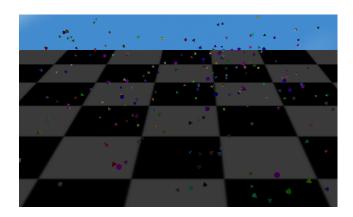


Figure 1: Example of evolving swarm and environment.

Berman et al., 2011). Flying swarms have also been applied in the lab to tasks such as chemical cloud detection (Kovacina et al., 2002) and dynamic communications relays (Hauert et al., 2008).

Evolving Swarms

One major concern of evolutionary biology, which has been studied by both biologists and computer scientists, is the evolution of collective behavior (such as group foraging and swimming in schools) among a group of organisms. Previous studies have examined the evolutionary risks and benefits of some of these behaviors, and have examined them as optimization processes that could be stable or unstable under different circumstances (Davies et al., 2012; Pulliam and Caraco, 1984; Sibly, 1983). Artificial life researchers have used computational systems to study the evolution of parasitism (Ray, 1991), the collective behavior of flying and swimming organisms (Reynolds, 1993; Zaera et al., 1996), and the interactions of evolution and game theory (Eriksson and Lindgren, 2002).

In recent years there has been some success in evolving collective behavior amongst flying artifical agents using SwarmEvolve and SwarmEvolve 2.0 Spector and Klein (2002); Spector et al. (2005), which modeled 3D virtual worlds and allowed for goal orientation, multiple species of

birds, and evolution of the motion control equation itself. This success has been followed by other work in evolving coordinated flying groups (Knoester and McKinley, 2011) as well as evolving other kinds of group behavior such as wolf-pack hunting (Muro et al., 2011).

Evolution and Environmental Variation

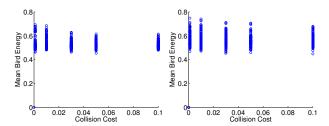
In the study of evolution and ecology, environmental variation is receiving increasing attention as a factor due to concerns about climate change. Ruel and Ayres (1999) used Jensen's inequality, a mathematical proof, to predict some effects of environmental variation on biological systems. Other studies have found that environmental variation affects the ability of species to coexist (Chesson, 1986) and that the more abrupt environmental variation of climate change has different effects on population dynamics than does natural environmental variation (Ruokolainen et al., 2009). In microbial ecosystems, the resilience of populations has been linked to the degree of environmental variation, where harsh variations can trigger population collapse (Sanchez and Gore, 2013). Several researchers (Visser and Both, 2005; Stenseth and Mysterud, 2002) have studied the potential effects of climate change on periodic animal and plant life cycle events, and the effects of such mistiming on food availability (Both, 2010). Recently, evolutionary game theory (Weibull, 1997) has been applied to questions of environmental variation, climate change, and ecology (Johansson and Jonzén, 2012a,b).

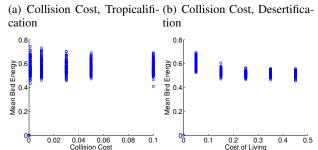
In this paper, we examine the effects of different forms of environmental variation on the evolution of a flying swarm.

Model

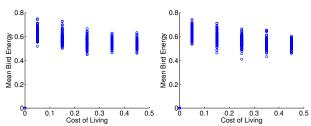
Simulation is performed with the Brevis simulator (Harrington, 2014), a scientific and artificial life simulator. Brevis provides simulation and visualization capabilities via the Java JVM and the programming language, Clojure. In addition to visualization, Brevis provides a number of simulation features including neighborhood and collision detection, both of which are key to 3D swarm simulations.

Our model consists of a population of birds with 6 continuous genetic traits and a set of foods (energy sources). The flight of a bird is controlled by its genes, similarly to (Reynolds, 1987). Each bird uses the first bird and first food in its list of neighbors, if there are any neighbors of either type within the neighborhood radius (10,000 units, in this case). The direction vectors between the bird and its neighboring food and bird are then computed. If the respective entity does not exist in the neighborhood a vector of magnitude 0 is used. While we have chosen to use a neighborhood size that is large enough to ensure that birds will always have another bird and food in its neighborhood, alternative default behaviors may be more favorable. The direction vectors are then weighted based upon whether the distance between the bird and the entity is "close" or "far,"





(c) Collision Cost, Rapid De- (d) Cost of Living, Tropicalifisertification cation



(e) Cost of Living, Desertifia- (f) Cost of Living, Rapid Detion sertification

Figure 2: Collision cost (a-c) and cost of living (d-f) vs mean bird energy in tropicalification, desertification, and rapid desertification simulations, with best-fit quadratic curves displayed.

where 2 distance genes determine this threshold for neighboring birds and foods. The sum of the weighted vectors is then taken as the bird's new acceleration vector. A planar floor is positioned in the world at y = 0, and when birds collide with the floor they "land," such that they lose their current velocity and acceleration, reorient to be perpendicular to the floor, and automatically push off with small initial velocity and acceleration.

Energy

Energy is the currency of our simulation. Birds require energy to live, and foods produce energy over time. Each bird is subject to a constant cost of living, which influences how long a bird can survive between feedings. The energy in foods are replenished a constant rate (0.1 units/timestep). Energy is transferred between entities when collisions occur. A collision between two birds results in an energy loss for both birds. Collisions between a bird and a food result in a transfer of energy from the food to the bird at a rate of

0.005 units/timestep. When a food's energy reaches zero, it is removed from the simulation and replaced by a newly initialized and randomly positioned food. If a bird's energy reaches zero, it is removed by the simulation and replaced. In our experiments we study a range of values for both the cost of living for birds, and the cost of collisions between birds.

Evolution

Each bird has a genome of 6 genes, which is used to control its flight acceleration. Two genes are distance thresholds that indicate whether an entity is considered close or far, for neighboring foods and neighboring birds. The remaining 4 genes are coefficients that specify the weights for the close/far neighboring birds/foods. Distance threshold genes are bounded by the hypotenuse of the area containing food (565.69 units), while the remaining acceleration weighting genes are within [-10, 10]. By using a fixed set of continuous traits, the genetic diversity in the population is mostly regulated by selection dynamics, whereas genetic diversity in models utilizing discrete traits is more vulnerable to loss of diversity via mutation.

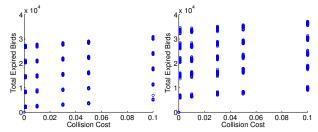
Every time a bird runs out of energy, it is replaced. The replacement is the mutant of a randomly selected living bird that has been alive for more than 1 timestep 99% of the time, and 1% of the time a completely randomly generated bird is used as the replacement. Mutation is achieved by adding/subtracting random values to each gene in the mutant. Birds better able to survive implicitly have more opportunities to reproduce.

Experiments

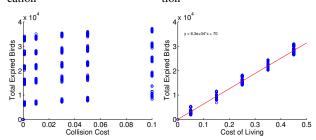
We ran simulations of 250 birds each over 100,000 timesteps¹. In each simulation, the birds randomly traversed a world containing scattered food items, where a bird gains energy from colliding with a food item. Birds were required to maintain a certain level of energy in order to survive, and could expire either from insufficient energy or from exceeding a maximum lifespan, with any expired bird being replaced by a new bird.

Different environments were represented by different food densities. The "uniform-high" environment, representing a tropical or other lush climate, was characterized by a uniformly high food density. The "uniform-low" environment, representing a sparsely-vegetated climate such as a desert or tundra, was characterized by a uniformly low food density. The "seasonal" environment, representing a temperate climate, had "summers" of high food density and "winters" of low food density.

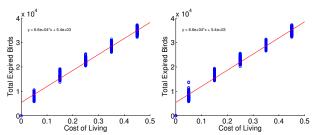
Each simulation switched from one environment to another at the halfway point, with three types of environmen-



(a) Collision Cost, Tropicalifi- (b) Collision Cost, Desertificacation tion



(c) Collision Cost, Rapid De- (d) Cost of Living, Tropicalifisertification cation



(e) Cost of Living, Desertifia- (f) Cost of Living, Rapid Detion sertification

Figure 3: Collision cost (a-c) and cost of living (d-f) vs total expired birds in tropicalification, desertification, and rapid desertification simulations, with best-fit lines displayed for d-f.

tal variation being tested. The first two types, uniform-high to uniform-low and seasonal to uniform-low, represent desertification of tropical and temperate environments, respectively. The third type, seasonal to uniform-high, represents the encroaching of tropical climates onto previously-temperate environments. We refer to these as "tropicalification," "desertification", and "rapid desertification."

For each type of environmental variation, we sampled across different values for two parameters, meant to approximate certain real-world ecosystem dynamics. The first, collision cost, takes away a certain amount of energy from a bird that collides with another bird, and is meant to approximate the effects of competition. The second, cost of living, is the amount of energy that a bird must take in over a certain time period in order to stay alive. Simulations for all three types of environmental variation were run with collision costs of 0.001, 0.01, 0.03, 0.05, and 0.1, and costs

¹Simulation code used for this study will be made publicly accessible upon completed documentation via the Brevis website: http://brevis.golemics.org

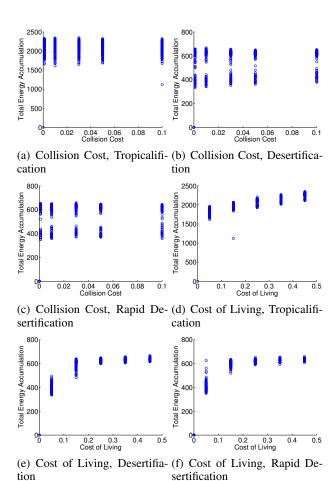


Figure 4: Collision cost (a-c) and cost of living (d-f) vs total accumulated energy in tropicalification, desertification, and rapid desertification simulations, with best-fit quadratic curves displayed for d-f.

of living of 0.05, 0.15, 0.25, 0.35, and 0.45. Each set of collision cost and cost of living parameters was used in 25 simulations.

Results and Discussion

We look at how our two parameters that mimic ecological dynamics, collision cost and cost of living, are associated with three different variables - mean bird energy, total expired birds during the last quarter of the simulation, and total accumulated energy during the last quarter of the simulation. We measured mean bird energy three quarters of the way through each simulation, so that we would see their values midway through the post-environmental-change period, once the birds had had some chance to adapt to the new environment. Total expired birds and total accumulated energy were measured for the last quarter of each simulation, so that the measurements would start after the birds had had the opportunity to adapt to the new environment and so that there would be enough time remaining for a cumulative metric to

be reasonable.

We first looked at how collision cost and cost of living are associated with mean bird energy. In Fig.2(a)-2(c), we can see how collision cost is associated with mean bird energy for all three types of environmental variations. The differences between tropicalification, desertification, and rapid desertification here are very subtle. Both types of desertification showed slightly wider ranges of mean bird energy across simulations with the same collision cost. However, increasing collision cost had minimal effect on mean bird energy.

The effect of cost of living on mean bird energy, shown in Fig. 2(d)-2(f), depended on what sort of environmental variation took place in the simulation. As with collision cost, both forms of desertification produced wider ranges of results for the same cost of living, suggesting that desertification causes more unpredictability regarding mean bird energy than tropicalification. In all simulations, mean bird energy fell as cost of living increased and birds needed to expend more energy in order to survive. However, in the tropicalification simulations, the decline in mean bird energy gradually leveled off as cost of living increased, while in both types of desertification simulations, this leveling off was less clear. This suggests that the birds may have been more resilient to increased energy needs when food became more plentiful rather than more scarce.

Next, we looked at how collision cost and cost of living influenced the total number of expired birds in the simulations. In both cases, the type of environmental variation - tropicalification vs desertification - made the biggest difference, with desertification of any kind leading to more expired birds.

When collision cost was correlated with total expired birds, as shown in Fig. 3(a)-3(c), we found that there was a small increase of expired birds as collision cost increased, and this was true for both tropicalification and desertification simulations. This is an intuitive result, as the steeper sudden drops in a bird's energy caused by higher collision cost would make it more likely to fall under the energy threshold needed for it to stay alive. Desertification situations, in which once-plentiful food became more scarce, also unsurprisingly led to more expired birds than the reverse situation. In the desertification simulations, the number of expired birds at the same collision cost was not only slightly higher but also more variable than in the tropicalification simulations, and rapid desertification was correlated with slightly more variability than slow desertification. If we recall that collision cost is meant to approximate competition for territory between birds, this result makes sense. As food becomes more sparsely distributed, more birds will need to feed from the same areas of the map. There is a tradeoff between being able to find food and being able to avoid collisions. In different simulations this tradeoff may play out with slightly different dynamics, leading to greater variability. Since the change in food geography is more drastic in rapid than in regular desertification, the effect is greater in rapid desertification. Total expired birds over different experimental runs could be grouped into five clusters, which correspond to the different costs of living in different runs, suggesting that cost of living plays a greater role in bird expirations than collision cost.

As with the previous figure, in Fig. 3(d)-3(f), we see that desertification of any sort increases the overall number of expired birds. Under all three types of environmental variance, we see that increases in the energy needed to live lead to increases in the number of expired birds, as the birds have a more difficult time maintaining the higher levels of energy. We also see that in the desertification simulations, the rate at which expired birds increase with cost of living increases, which can clearly be seen with the increased slopes (P = 0.032 and P = 0.036) of the best-fit lines in the desertification graphs as compared to the tropicalification graph. However, there is little difference between the two types of desertification - going from a high-energy environment to a low-energy one seems to be just as much of a problem in this case as going from a seasonal, temperate environment to a low-energy one.

Finally, we looked at how our varying parameters influenced the total accumulated energy in the simulations. Unsurprisingly, it was much higher overall - by more than a factor of three - in the tropicalification simulations where food started out adequate and became plentiful, regardless of these parameters. In Fig. 4(a)-4(c), we can see that collision cost had little influence on total accumulated energy, but desertification was connected to bimodal levels of total accumulated energy at low and moderate collision costs, with some simulations accumulating a high level, and some at the same collision cost accumulating a low level. This may be an indication that sometimes, under these collision costs, the birds were able to evolve reasonably quickly to be able to find the sparser food, and sometimes they were not. Again, there is a potential tradeoff between being able to find the food and being able to avoid collisions, as the same number of birds feeds from fewer food sources. The observed bimodality was less strong with the regular desertification than with the rapid desertification, indicating a gradual slide into bimodality as the intensity of environmental variation increases.

In Fig. 4(d)-4(f), the plots of the two different types of desertification simulations against energy cost of living are similar to each other, but under rapid desertification, there was slightly more variance (P=0.0013) in total accumulated energy, especially at low cost of living. This may be because of the greater instability in the system introduced by the more rapid change.

Overall, desertification vs tropicalification had a greater impact than degree of desertification across the board, with the results from the two different types of desertification being very similar.

Evolution of Genes

We also tracked the effects of environmental variation on our swarms at the genetic level. This is depicted for three simulations, one of each environmental variation type, in Figs. 5(a)-5(c). Each bird has 6 evolving genes to control flight acceleration, which are described in more detail in the "Model" section above. The distance genes neighborD and foodD set the thresholds for how close a neighboring bird or food source, respectively, has to be to the bird to be considered "close" vs "far." The other four genes specify weights for close/far neighboring birds and foods. In this way, neighbor acceleration genes (neighborC and neighborF) assist in collision avoidance while food acceleration genes (foodC and foodF) assist in finding food.

Under conditions of rapid desertification (with the environmental variation taking place halfway through, as in all the simulations), the weights of neighbor acceleration genes go quickly to zero or near zero, flatlining by the time the simulation is 70% complete. Under regular desertification this process is slower, not completing until more than 80% of the way through the simulation, and under tropicalification it does not quite happen at all. This suggests that a switch from a more food-rich to a food-scarce environment particularly selects for birds that seek out neighbors rather than avoiding them, as the presence of other birds can mean the presence of food, but it also suggests that this is a useful trait in general (particularly at such a low collision cost), as the magnitudes of weights of the neighbor acceleration genes decreased significantly even in tropicalification.

In the tropicalification simulation, the thresholds for considering a neighboring bird "close" as set by neighbor distance genes increased over time, while those of food distance genes decreased. This was not true in either of the desertification simulations - while the distance thresholds fluctuated after rapid desertification, apparently unable to stabilize after the shock to the system, their averages over time stayed nearly level, and during regular desertification it was the food distance thresholds that gradually increased. The fact that food was easier to find in the "tropics" may have meant that there was no need for relatively faraway foods to be considered "close" and easily detectable for birds, while with neighbors more spead out rather than flocking to the same food sources, there was evolutionary selection pressure to be able to detect and avoid colliding with less-predictablylocated neighbors. In desertification the reverse pressure would naturally be exerted for food, though it is is interesting that in the more intense desertification situation, in which the birds were adjusting to a larger change, they were unable to evolve this threshold pattern over the time given by the simulation.

Unsurprisingly since our modeling of environmental variation centered around food availability, food acceleration

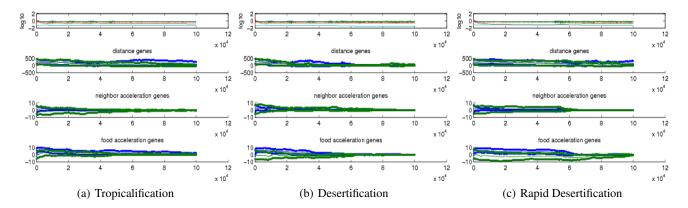


Figure 5: CoL = 0.05, CC = 0.001. In the top subplots, blue is the mean energy per bird, green is the mean energy per food, red is the bird expiration rate, and turquoise is the mean accumulated energy per bird. In the distance genes suplots, blue represents the neighborD gene and green the foodD gene +/- a standard deviation. In the neighbor acceleration genes subplots, blue represents neighborC, and green, neighborF, +/- a standard deviation. In the food acceleration genes subplots, blue represents foodC, and green, foodF, +/- a standard deviation.

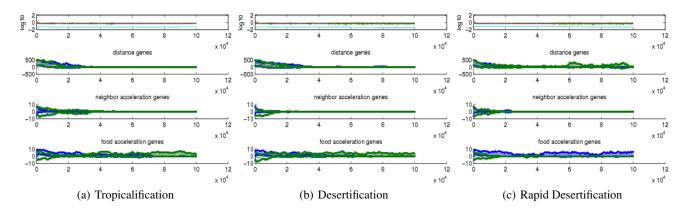


Figure 6: CoL = 0.15, CC = 0.15. In the top subplots, blue is the mean energy per bird, green is the mean energy per food, red is the bird expiration rate, and turquoise is the mean accumulated energy per bird. In the distance genes suplots, blue represents the neighborD gene and green the foodD gene +/- a standard deviation. In the neighbor acceleration genes subplots, blue represents neighborC, and green, neighborF, +/- a standard deviation. In the food acceleration genes subplots, blue represents foodC, and green, foodF, +/- a standard deviation.

genes were notably influenced by type of environmental variation. Under regular desertification, the weight magnitudes of the foodF genes go nearly to zero, and there is some slight maintenance of the weight magnitudes of the foodC ones. Under rapid desertification, all food acceleration genes had high (though gradually decreasing) weight magnitudes when food was plentiful, but those of foodC genes go nearly to zero while those of foodF genes maintain better - the opposite of the tropicalification simulation, in which the weight magnitudes of foodF genes go nearly to zero while those of foodC genes maintain better. This makes sense, as in the former case, food is more likely to be far away from any given point in the world, while in the latter, food is likely to be nearby and there is less need to be able to deal productively with faraway food.

For comparison, we look at the results of simulations with a higher collision cost and cost of living (both 0.15), which are visible in Figs. 6(a)-6(c). In these, we see the weight magnitudes of neighbor acceleration genes neighborC and neighborF going nearly to zero in all three simulations even before the environmental variation - even with the strongly increased collision cost, the increased cost of living appears to make finding food more important than being able to avoid neighbors. Distance genes foodD and neighborD, whose thresholds go to zero in the other simulations, fluctuate after rapid desertification, especially foodD - while in the slower cases, the birds are adapting relatively quickly to new needs around food, the larger perturbation of food availability in the rapid desertification scenario appars to make these genetics unstable, as also happened with lower collision cost

and cost of living.

The food acceleration genes show an intriguing pattern - in both simulations where the environment starts out seasonal, the weight magnitudes of the foodF genes quickly increase (and do so further after the environmental change), but in the rapid desertification simulation the foodC genes start out with higher weight magnitudes that start to decrease before increasing again but fluctuating after desertifiction. One might assume that this is because the rapid desertification scenario is the only one in which food started out at its most plentiful, making it less necessary to need to seek out faraway food and more necessary to seek out close food, however, this does not explain why this was still the case after desertification. Once again, as with the distance genes in the same scenario, the rapid, intense environmental variation is connected with a lack of genetic stability that is not seen with the other forms of environmental variation. Indeed, the post-change lack of genetic stability in the population seems to be a characteristic of the rapid desertification scenario at these moderate collision cost and cost of living parameters, and is even seen to a lesser extent using the more survivalfriendly parameters discussed previously. In all cases food acceleration genes play a much greater role than when cost of living was lower.

We have simulated a swarm of birds evolving genetically and behaviorally to three different types of environmental variation, meant to approximate types of variation seen in the real world during times of climate change - a temperate seasonal environment to a tropical one, a seasonal enviroment to a desert, and a tropical environment to a desert. We found that desertification in particular led to negative outcomes such as increased expired birds and decreased population stability (as indicated by the amount of variation on such metrics as mean bird energy or total accumulated energy between simulations that used the same parameters). The direction of the variation - tropicalification or desertification - had a greater impact on these outcomes than the intensity of the variation (seasonal to desert vs tropical to desert). The environmental variation also affected the genetics of the birds, with decreased food availability leading to selection against collision avoidance genes, and food availability influencing whether bird ability to find nearby food or faraway food was favored. A greater intensity of environmental variation led to less stability in population genetics post-environmental shift. In the wake of concerns about climate change, it is increasingly important to be able to predict how populations will fare under environmental variation. As Sanchez and Gore (2013) found that harsher environmental variation led to decreased resilience in microbial populations, we have found that it led to decreased genetic resilence in our simulated birds. In addition, we have identified the type rather than just the intensity of variation as another factor influencing resilience.

Acknowledgements

This work was partially supported by NSF grant No. 1068620.

References

- Berman, S., Kumar, V., and Nagpal, R. (2011). Design of control policies for spatially inhomogeneous robot swarms with application to commercial pollination. In *Robotics and Automation (ICRA)*, 2011 IEEE International Conference on, pages 378–385. IEEE.
- Both, C. (2010). Food availability, mistiming, and climatic change. *Effects of climate change on birds. Oxford University Press, Oxford*, pages 129–147.
- Chesson, P. L. (1986). Environmental variation and the coexistence of species. *Community ecology*, 240:54.
- Davies, N. B., Krebs, J. R., and West, S. A. (2012). *An introduction to behavioural ecology*. John Wiley & Sons.
- Eriksson, A. and Lindgren, K. (2002). Cooperation in an unpredictable environment. In *Proc. Eighth Intl. Conf. on Artificial Life, The MIT Press: Cambridge, MA*, pages 394–399.
- Harrington, K. I. (2014). Brevis (version 0.7.3).
- Hauert, S., Winkler, L., Zufferey, J.-C., and Floreano, D. (2008). Ant-based swarming with positionless micro air vehicles for communication relay. Swarm Intelligence, 2(2-4):167–188.
- Johansson, J. and Jonzén, N. (2012a). Effects of territory competition and climate change on timing of arrival to breeding grounds: a game-theory approach. *The American naturalist*, 179(4):463–474.
- Johansson, J. and Jonzén, N. (2012b). Game theory sheds new light on ecological responses to current climate change when phenology is historically mismatched. *Ecology letters*, 15(8):881–888.
- Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical report, Erciyes university, engineering faculty, computer engineering department.
- Knoester, D. B. and McKinley, P. K. (2011). Evolving virtual fireflies. In Advances in Artificial Life: Darwin Meets von Neumann, pages 474–481. Springer.
- Kovacina, M. A., Palmer, D., Yang, G., and Vaidyanathan, R. (2002). Multi-agent control algorithms for chemical cloud detection and mapping using unmanned air vehicles. In *Intelligent Robots and Systems*, 2002. IEEE/RSJ International Conference on, volume 3, pages 2782–2788. IEEE.
- Lardon, L. A., Merkey, B. V., Martins, S., Dötsch, A., Picioreanu, C., Kreft, J.-U., and Smets, B. F. (2011). idynomics: nextgeneration individual-based modelling of biofilms. *Environmental microbiology*, 13(9):2416–2434.
- Mach, R. and Schweitzer, F. (2003). Multi-agent model of biological swarming. In *Advances in Artificial Life*, pages 810–820. Springer.
- Muro, C., Escobedo, R., Spector, L., and Coppinger, R. (2011). Wolf-pack (i i canis lupusi/i) hunting strategies emerge from simple rules in computational simulations. *Behavioural processes*, 88(3):192–197.

- Nagpal, R., Berman, S., and Halász, Á. (2011). Optimization of stochastic strategies for spatially inhomogeneous robot swarms: A case study in commercial pollination.
- Perez, L. and Dragicevic, S. (2010). Modeling mountain pine beetle infestation with an agent-based approach at two spatial scales. *Environmental modelling & software*, 25(2):223–236.
- Pulliam, H. R. and Caraco, T. (1984). Living in groups: is there an optimal group size. *Behavioural ecology: an evolutionary* approach, 2:122–147.
- Ray, T. S. (1991). Is it alive or is it {GA}. In Proceedings of the Fourth International Conference on Genetic Algorithms, pages 527–534. Morgan Kaufmann.
- Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of SIGGRAPH Computer Graphics*, volume 21, pages 25–34. ACM.
- Reynolds, C. W. (1993). An evolved, vision-based behavioral model of coordinated group motion. From animals to animats, 2:384–392.
- Rubenstein, M., Ahler, C., and Nagpal, R. (2012). Kilobot: A low cost scalable robot system for collective behaviors. In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pages 3293–3298. IEEE.
- Ruel, J. J. and Ayres, M. P. (1999). Jensen's inequality predicts effects of environmental variation. *Trends in Ecology & Evolution*, 14(9):361–366.
- Ruokolainen, L., Lindén, A., Kaitala, V., and Fowler, M. S. (2009). Ecological and evolutionary dynamics under coloured environmental variation. *Trends in Ecology & Evolution*, 24(10):555–563.
- Sanchez, A. and Gore, J. (2013). Feedback between population and evolutionary dynamics determines the fate of social microbial populations. *PLoS biology*, 11(4):e1001547.
- Sibly, R. (1983). Optimal group size is unstable. *Animal Behaviour*, 31(3):947–948.
- Spector, L. and Klein, J. (2002). Evolutionary dynamics discovered via visualization in the breve simulation environment. In Workshop Proceedings of the 8th International Conference on the Simulation and Synthesis of Living Systems, pages 163–170.
- Spector, L., Klein, J., Perry, C., and Feinstein, M. (2005). Emergence of collective behavior in evolving populations of flying agents. *Genetic Programming and Evolvable Machines*, 6(1):111–125.
- Stenseth, N. C. and Mysterud, A. (2002). Climate, changing phenology, and other life history traits: nonlinearity and match-mismatch to the environment. *Proceedings of the National Academy of Sciences*, 99(21):13379–13381.
- Su, H., Wang, X., and Lin, Z. (2009). Flocking of multi-agents with a virtual leader. *Automatic Control, IEEE Transactions* on, 54(2):293–307.
- Visser, M. E. and Both, C. (2005). Shifts in phenology due to global climate change: the need for a yardstick. *Proceedings of the Royal Society B: Biological Sciences*, 272(1581):2561–2569.

- Weibull, J. W. (1997). Evolutionary Game Theory. MIT press.
- Zaera, N., Cliff, D., et al. (1996). (not) evolving collective behaviours in synthetic fish. In *Proceedings of International Conference on the Simulation of Adaptive Behavior*.

Real-time Evolution of iAnt Robot Foraging Strategies

Joshua P. Hecker¹ and Melanie E. Moses^{1,2,3}

¹Department of Computer Science, University of New Mexico, Albuquerque, NM 87131

²Department of Biology, University of New Mexico, Albuquerque, NM 87131

³External Faculty, Santa Fe Institute, Santa Fe, NM 87501

{jhecker,melaniem}@cs.unm.edu

Introduction

Central-place foraging is a canonical task in swarm robotics. For this task, robots are programmed to search an area for resources and aggregate these resources at a central location. Foraging can be instantiated in a number of real-world applications, such as hazardous waste clean-up, search and rescue, and in-situ resource utilization.

We extend our prior work by using coevolutionary methods to evolve iAnt robot swarm foraging strategies in real robots and in real time. Each robot maintains a private agent-based simulation, which models the physical environment according to the robot's own approximation of the realworld resource distribution. The robot uses a private genetic algorithm (GA) to evolve parameters for a central-place foraging algorithm (CPFA) which maximize the foraging success of the simulated agents; it then employs these evolved parameters as a foraging strategy in the real world.

In addition to evolving individual foraging strategies, the entire swarm executes a distributed GA to evolve a population of resource distribution approximations that is shared across all robots. In this way, coevolution evolves both the foraging strategy and the resource distribution approximation for each robot. Over time, this allows the swarm to adapt its behavior to previously unknown environments.

Background

A key challenge in evolutionary robotics is narrowing the reality gap between simulated agents and real robots. Bongard and Lipson (2004) address this by iteratively adapting robots and robot simulators in real time to resolve brain-body discrepancies and thus generate robust machines. O'Dowd et al. (2011) use distributed coevolution to coevolve their foraging strategy in conjunction with their simulated world model in order to adapt to different simple foraging tasks.

Our approach differs from this previous work in that we use interdependent foraging strategies, memory, and communication to accomplish more complex foraging tasks. In this paper, we adapt foraging behaviors to maximize foraging success on the specific resource distribution that the robots encounter in each foraging experiment.

Methods

The CPFA implements robot foraging behaviors as a series of states connected by directed edges with transition probabilities (Fig. 1(a)). Each robot begins its search at a central nest site and sets a search location. Robots traveling to a random location with no prior information search using an uniformed correlated random walk. Robots traveling to a previously found resource location via memory or communication search using an informed random walk that is initially undirected and localized, then becomes more directed and straighter over time. When a robot locates a resource, it first collects the resource, and then records a count of resources in the neighborhood of the found resource. Robots use this count to decide whether to exploit information through memory or communication. Robots who have not found a resource will probabilistically return to the nest.

In prior work, we used a GA to evolve a population of CPFA parameters that maximized the foraging efficiency of simulated robot swarms evaluated in an agent-based model (Hecker et al., 2013). These parameters control the sensitivity threshold for triggering CPFA behaviors, the likelihood of transitioning from one behavior to another, and the length of time each behavior should last. The fitness function of the GA is the collective foraging success (number of resources collected in fixed time). Each parameter set is evaluated with an identical copy running on each robot. We demonstrated that such "group selection" evolved successful foraging strategies for each resource distribution, and that robot swarms were most efficient when using the specialist strategy adapted for a given distribution. However, the approach required a priori knowledge of the resource distribution, and it did not evolve in real time.

In this work, we simulate individual robots, each running a private version of the GA in real time while foraging given a particular unspecified resource distribution (Fig. 1(b), point 1). The parameter set evolved by the private GA is then transferred to the robot and evaluated for fitness (Fig. 1(b), point 2). Each robot defines its fitness as the number of actual resources it individually collects during the experiment (Fig. 1(b), point 3). Robots periodically communicate

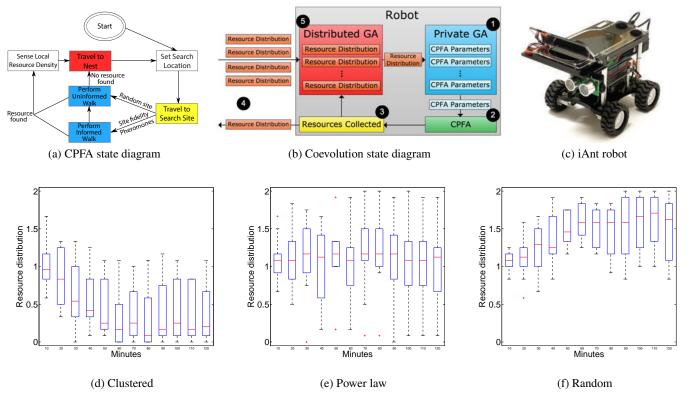


Figure 1: Top: Diagrams explaining implementation of (a) CPFA and (b) coevolution, as well as (c) an iAnt robot. Bottom: Resource distribution approximation values for swarms foraging on (d) clustered, (e) power law, and (f) random distributions.

their current foraging efficiency and simulation resource description to the entire swarm via wireless transmission (Fig. 1(b), point 4). Each robot evolves a distinct subpopulation of simulated resource placements (Fig. 1(b), point 5). In this way, the swarm's distributed GA maximizes the correspondence between agent-based simulations and the resource distribution by selecting for simulations which produce more efficient foraging strategies in real robots (Fig. 1(c)).

Preliminary Experiments

We conduct preliminary experiments with simulated robot swarms in an agent-based model. Swarms of 12 robot agents forage for 256 resources placed on a 125 x 125 cellular grid, simulating a 100 m^2 physical area over 120 minutes. The resources are arranged in one of three distributions: clustered (4 randomly placed clusters of 64 resources each), power law (1 large cluster of 64, 4 medium clusters of 16, 16 small clusters of 4, and 64 randomly scattered), or random (each resource placed at a random location).

Each robot's private simulation (Fig. 1(b), point 1) is randomly initialized with one of the three distributions. The CPFA parameters of each swarm in the private simulation are randomly and independently initialized; agents within a swarm use identical parameters. The robot's private GA evolves a population of 100 simulated swarms over 20 generations. When the private simulation is complete (approx-

imately 10 minutes), the robot transfers the most efficient CPFA parameters from the simulation to its own CPFA (Fig. 1(b), point 2). Each robot then communicates to the entire swarm its current foraging efficiency, along with a value representing the current resource distribution used in its private simulation (clustered, power law, or random).

Figures 1(d)–(f) show the resource distribution approximation values for the entire swarm at 10 minute increments across 10 replicates. The swarm converges on the true distribution in each experiment (clustered = 0, power law = 1, random = 2) within approximately 60 minutes. Adapting foraging behavior to a particular resource distribution in real time is 2.7 times more efficient for clustered resources, and 1.1 times more efficient for random resources, compared to using a generalist fixed strategy evolved *a priori* for power-law-distributed resources. Experiments replicating these observations in real iAnt robots are ongoing.

References

Bongard, J. C. and Lipson, H. (2004). Once More Unto the Breach: Co-evolving a robot and its simulator. In *ALIFE9*, pp. 57–62.

Hecker, J. P., Stolleis, K., Swenson, B., Letendre, K., and Moses, M. E. (2013). Evolving Error Tolerance in Biologically-Inspired iAnt Robots. In ECAL 2013, pp. 1025–1032.

O'Dowd, P. J., Winfield, A. F., and Studley, M. (2011). The distributed co-evolution of an embodied simulator and controller for swarm robot behaviours. In *IROS* 2011, pp. 4995–5000.

Don't Believe Everything You Hear; Preserving Relevant Information by Discarding Social Information

Christoph Salge¹ and Daniel Polani ¹

¹University of Hertfordshire, Hatfield, UK c.salge|d.polani@herts.ac.uk

Abstract

Integrating information gained by observing others via Social Bayesian Learning can be beneficial for an agent's performance, but can also enable population wide information cascades that perpetuate false beliefs through the agent population. We show how agents can influence the observation network by changing their probability of observing others, and demonstrate the existence of a population-wide equilibrium, where the advantages and disadvantages of the Social Bayesian update are balanced. We also use the formalism of relevant information to illustrate how negative information cascades are characterized by processing increasing amounts of non-relevant information.

Introduction

Information processing is an important aspect of life. Organisms equipped with sensors obtain and utilize information to increase their inclusive fitness; thus justifying the existence of (often costly) sensors in the first place (Polani, 2009). However, not all information is equally relevant for an organism – a notion formalised by Polani et al. (2001, 2006), which we will introduce in more detail later. The basic idea of *relevant information* is to quantify how much information at least is needed to obtain a certain performance level. Once this is established, the next question to ask is, how to best obtain this specific information?

Previously, we argued (Salge and Polani, 2011) that agents with common goals and embodiments are likely to have similar relevant information. Once they obtain this relevant information, they also have to act upon it to reap its benefits, thus encoding it in their actions. As the state-space of actions is usually much smaller than the state-space of the overall environment, this is likely to lead to a higher "concentration" of relevant information in another agent's actions rather than in the environment itself. This digested information, encoded in actions, concentrates pre-processed decision-relevant information and provides incentives for agents to observe each other and modify their own actions accordingly. However, similar behaviour in a population of agents can lead to a phenomenon called herding (Banerjee,

1992) or *information cascade* (Bikhchandani et al., 1992). This usually requires an agent population where agents:

- select one of several choices;
- have some private information related to their decision;
- act sequentially and can observe the choices of others, but not the private internal information of others.

This can then lead to situations such as the example by Easley and Kleinberg (2010), where an agent wants to choose between restaurant A and B. His own research suggests that restaurant A is better, but once he gets there, no one is eating in restaurant A, while restaurant B is filled with customers. Based on this information it is reasonable to infer that several other agents have private information that caused them to choose B instead of A. By inferring this additional information it becomes rational to choose B instead of A, even if his own private information suggests otherwise.

The problem here is that others might make similar conclusions, and create a chain reaction of inferred private information that is based on no or very little private information. This illustrates two common properties of information cascades; they can be based on very little initial information, and they can be wrong.

This is somewhat in contrast to the argument presented in "The Wisdom of Crowds", where Surowiecki (2005) argues that agents that aggregate their information can produce very accurate results. But, as Easley and Kleinberg (2010) point out, this only applies if they are guessing independently. Furthermore, recent studies (Kao and Couzin, 2014) examining several models of group behaviour suggest that small groups make correct decision, while larger groups are more likely to converge on an incorrect decision. Also note that information cascades are also present in other types of multi-agent scenarios, such as swarm coordination (Wang et al., 2012), and are potentially subject to similar problems.

Overview

In this paper we examine the interaction between the positive and negative effects of observing others through the perspective of the relevant information framework. In particular, we show how rational adaptations can lead to a situation where incorrect information cascades become common, and how they are characterized by a reduction in the density of relevant information. Furthermore, we demonstrate that in this environment it is reasonable for agents to randomly discard part of their sensor intake.

After introducing information theory and relevant information in more detail, we present the single agent model to create a baseline for agent performance and demonstrate how an agent's actions encode information. The multi-agent scenario is then used to motivate the introduction of the Social Bayesian Update, as it demonstrates the increase in performance when information from other agents is used in decision making. The next scenario deals with changing world states and shows that agent's performance can be increased by explicitly modelling the noise in the world, which basically motivates internal models which cannot express certainty. This specific form of bounded rationality is interesting in the context of information cascades, as Acemoglu et al. (2011) previously showed that a lack of internal certainty makes populations more likely to synchronize. Finally, we will look at models that combine noise and Social Bayesian Update, which have both been motivated previously by increased agent performance. In this environment, negative information cascades are common but we show that agents can randomly discard sensor inputs to increase their performance. This is motivated by results from Gale and Kariv (2003), which demonstrated that sparsity in the observation graph makes convergence (both negative and positive) less likely. By moderating their own sensor intake, agents can change between single-agent behaviour, and positive "wisdom of the crowds" and negative information cascades.

Information Theory

Relevant information is based on the formalism of Information Theory (Shannon, 1948). If X is a random variable that can assume the states x, where each state x is a member of the alphabet \mathcal{X} , then P(X) is the probability distribution of X, and P(X=x) is the probability that X assumes the value x, sometimes shortened to p(x). Entropy, or the self-information of a variable is then defined as

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x). \tag{1}$$

This is often described as the uncertainty about the outcome of X, the average expected surprise, or the average information gained if one was to observe the state of X, without having prior knowledge about X. Consider two jointly distributed random variables, X and Y; then we can calculate the *conditional entropy* of X given a particular outcome Y = y as

$$H(X|Y=y) = -\sum_{x \in \mathcal{X}} p(x|y) \log p(x|y). \tag{2}$$

This can be averaged over all states of Y, resulting in the conditional entropy of X given Y,

$$H(X|Y) = -\sum_{y \in \mathcal{Y}} p(y) \sum_{x \in \mathcal{X}} p(x|y) \log(p(x|y)).$$
 (3)

This is the entropy of X that remains, on average, if Y is known. So H(X) and H(X|Y) are the entropy of X before and after we learn the state of Y. Thus, their difference is the amount of information we can learn, on average, about X by knowing Y. Subtracting one from the other, we get a value called *mutual information*:

$$I(X;Y) = H(X) - H(X|Y).$$
 (4)

The mutual information is symmetrical and measures the amount of information one random variable contains about another (and vice versa, by symmetry). Also, note that we use the binary logarithm for all log(.) operations, so all information measurements are in *bits*.

Relevant Information

Relevant information is the amount of information an agent needs to obtain to either act optimally, or at a specific performance level. Assume that there is an agent that interacts with the environment by choosing an action in reaction to some form of sensor input. The environment R is in the state r, and the agent chooses an action a from a set of actions A. For simplicity, we assume for now that the agent can perceive the whole environment, so the sensor state is equal to the state of the environment. Furthermore, assume that the actions of the agent are connected to some utility function U(a,r) (for example, survival probability, or fitness) which determines different pay-offs, depending on the agent's action A=a and the state of the environment R=r. We also assume that the states of the world R are distributed according to the probability distribution P(R).

A *strategy* is defined as a conditional probability distribution P(A|R), which defines for every state r the probability of choosing different actions a. We can define a set π^u as the set of all strategies that have the average pay-off level, or performance, of at least u as

$$\pi^{u} = \left\{ P(A|R) \middle| \sum_{a} \sum_{r} U(a,r) p(a|r) p(r) \ge u \right\}. \quad (5)$$

As a strategy P(A|R) also implies a distribution P(A) = P(A|R)P(R), we can compute the mutual information I(A;R) for each strategy. The relevant information for a specific performance level is then defined as

$$RI(u) := \min_{p(a|r) \in \pi^u} I(A; R), \tag{6}$$

which is the minimal mutual information over all strategies that achieve at least the average pay-off of u. As the mutual

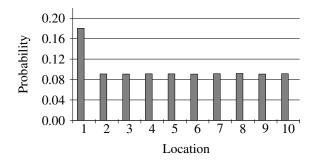


Figure 1: The probability of observing an agent going to a specific location, if the treasure is located in position 1 and there are 10 locations.

information I(A;R) measures the amount of information the agent has to process to determine a, this can be interpreted as the minimal amount of information an agent needs to obtain to perform at least as well as u. Due to the symmetry of mutual information, this can also be interpreted as the minimal amount of information an agent's actions have to contain.

Experiments

Single Agent Model

There are ten locations; exactly one of them contains treasure. The treasure location is modelled by the state of the variable T. The agent's task is to determine the location of the treasure in the least number of turns. Each turn the agent decides to visit one of the locations, and is then informed if that location contains the treasure or not.

The agent's decision making is modelled with an internal Bayesian model \hat{T} , where $P(\hat{T} = t)$ is the agent-assumed probability that the treasure is in location t. Every turn, the agent chooses to visit the location where it believes the treasure most likely to be. In case of a tie between different locations, it chooses one of them at random. Initially, the agent believes all locations to be equally likely. Once it observes the state of a given location, it updates its internal model with that knowledge. So, if location t is found empty, then it sets $P(\hat{T} = t) = 0$, and all other probabilities are uniformly scaled, so they still sum to one. If the agent finds the treasure, it is retired from the simulation. For this simple case the Bayesian model is not strictly necessary, but it will allow us to smoothly integrate later modifications. Here it just prevents the agent from revisiting any empty locations, which is arguably the best possible performance for an agent without any additional information.

For a world with ten locations it takes on average ≈ 5.5 turns to find the location with the treasure. This outperforms an agent which randomly visits (and revisits) locations (10 turns on average to find the treasure), which indicates that the agent is indeed processing information, and subsequently the agent's actions should contain relevant in-

formation. Fig. 1 shows the action distribution, gathered by observing 100,000 actions from different non-social agents while the treasure is in position 1. Note that none of the agents act after they found the treasure location, so all observed agents in Fig. 1 are ignorant of where the treasure is, but they know several location where it is not. This is enough processed information to imbue the agent's actions with relevant information.

To model the information another agent would acquire from observing one action from one randomly chosen agent, we assume that all agents are indistinguishable to an observer. If we model their action distribution with a variable called A, we can then use that data to compute how much information about T, the treasure location, is encoded in A. The mutual information in this case computes to ≈ 0.042 bit. We can compare those values to the information gained from observing a random location, which is ≈ 0.468 bit. So, while inspecting a location contains more information, observing another agent could provide additional information to enhance an agent's performance.

Performance is measured as the ratio of discovered treasure vs. turns. So, if an agent finds treasure on average once every five turns, it then has a performance ratio of 0.2. The single agent has a performance ratio of 0.180. This measurement is also identical to the fraction of agent actions that are looking at the right location. This allows us later to evaluate the performance of an agent population, as we do not have to measure the search time, but just measure how many of the agent's actions are going to the treasure location.

Multiple Agent Scenario

The last section indicated that the agent's actions contain relevant information about the treasure location. Therefore, we will now modify the model, so that the agent can integrate data from observing other agents into their internal belief model.

In the multi-agent model social agents will be able to observe the actions taken by other agents, but they will not see the result of this exploration, i.e. know if the visited location is empty. When an agent observes another agent's action a=A, it will integrate the obtained information into its own internal model $P(\hat{T})$ by performing a Naive Bayesian Update, based on the statistics for P(A|T) gathered from the non-social statistics in (Fig. 1). So, its new internal model after observing a is

$$P(\hat{T}|A=a) = \frac{P(A=a|T)}{P(A=a)}P(\hat{T}).$$
 (7)

If an agent finds the treasure, it will be replaced by a new agent, which is simulated by re-initializing an agent's internal model with the uniform distribution.

So, for the multi-agent simulation, all agents start with uniform internal distribution. Each turn the agents then decide their actions, based on their internal model, in the same sequential order. When agents observe other agent's actions, they update their internal model immediately. When agents observe a location, they either update their model if that location is empty, or are replaced by a new agent (have their model reset) if the location contains the treasure.

Note that the Naive Bayesian Update (NBU) works with the assumption that the different sources of information are independently distributed, which is not true in general. NBU still provides good approximations if the dependencies are normally distributed, but in information cascades this is also not the case, as the spread of information through a population is usually self-reinforcing. We still use the NBU, as a more exact Bayesian Update would be nearly impossible to produce, as it would require the agent to remember all previous interactions, and requires statistics on how all other sources of information interact. NBU on the other hand can be done the moment some information becomes available, and the internal belief representation can be represented as a single probability distribution.

Single Social Agent In the first experiment we examined 10 agents in a world with 10 locations. All data discussed from here on is the average value for 1,000 simulations, each running for 1,000 turns. Only *one* of the agents has the ability to observe the others. The location of the treasure is fixed, and determined at random at the beginning of the simulation. Unsurprisingly, the remaining non-social agents perform exactly as in the single agent simulation. Their distribution of actions matches the one recorded in Fig. 1.

The social agent in the simulation performs better; reaching a performance of ≈ 0.30 . This agent benefits from the information the other agents gather. As discussed in the "Digested Information" argument, the other agents act as information preprocessors for the social agent. Also, note that the distribution of actions of the social agent is even more concentrated on the actual treasure location, hence the mutual information between its actions and the treasure location, I(A;T)=0.220 bits, is higher than the same mutual information for the non-social agents, which was 0.042 bits.

All Social Agents Given the increase in performance for a single agent, we now assume that the whole population of agents adopts the social update approach, and we examine a simulation where all agents integrate the information gained from other agent's actions. This turns out to be extremely beneficial. The performance of the overall population, which is also the performance of every separate agent, is ≈ 0.99 . Once the treasure has been located by one agent, all subsequent actions lead to the treasure, and the mutual information between actions and treasure location is nearly maximal, $I(A;T) \approx \log(10)$.

Basically, the relevant information that the treasure is in location t propagates through the agents. It is displayed in an agent's actions, then used to update another agent's in-

ternal model. That agent then uses the information to determine which action to take, which is going to be A = t. The agent will then find the treasure and reset its internal model. But it will perceive others before it has to act again, biasing its internal model again towards taking action A = t. This will continue unless environmental information conflicts with this information, meaning the agent will not find the treasure at the location in which it was looking. In that case, the observed location's probability to contain treasure is set to zero, and the agent will look at other locations. This will initially get the agents to explore all locations until they find the treasure, after which they will all copy each other, finding the treasure every turn from that point onwards. Note, that the treasure does not move when it is found, however the agent who found the treasure resets its internal model (simulating its replacement with a new agent).

As we see, the important information is preserved by continuously flowing through the agent population. Even when agents retire and are replaced, the information is not lost. This looks like a very desirable feature for an agent population, and therefore the Social Bayesian Update seems like a reasonable adaptation.

Changing World State

In this section, we will demonstrate how lack of certainty can affect this simulation. We will use the single agent model to motivate the inclusion of noise into our internal Bayesian belief model.

In the next simulation the locations of the treasure will change during the simulation to different random locations. This will happen every turn with probability of P(change)=0.01. On average this should change the location every 100 turns. The behaviour of the agents is left unchanged.

First, let's again take a look at the simulation for a single agent. The performance ratio of the agent drops from 0.18 for the static world state simulation, to 0.14 for the simulation where the world state changes. A closer analysis shows that the agent's original behaviour has problems dealing with the new scenario. Consider that the agent visits a location x, and finds it empty. Then the probability for T=x will be set to zero in T. If the location now changes to T = x after the agent visited x, then the agent will first explore all other locations, finding all of them empty. This, in itself, is not problematic. But once the agent has looked at each locations once, all probabilities are assumed to be zero, given that the agent still assumes there is one, nonmoving treasure location. This is inconsistent with the basic properties of probabilities and is a result of the incorrect assumption about the immovability of the treasure location. In this specific implementation the agent now resorts to random search. This behaviour has, as we have seen, a lower performance rate, and therefore lowers the agent's overall performance.

Modelling Uncertainty To address this problem we can change the internal model to correctly reflect probabilities from the agent's perspective. The treasure changes its location with a probability of P(change) = 0.01 and relocates to one of the 10 locations randomly. This can be modelled by assuming that the world is in one of two states. Either, with P(change) = 0.01, it is in a state where the location has just changed, so T should be uniformly distributed with every $t \in T$ having the probability P(T = t) = 1/10. The other state, with a probability of 1 - P(change), is the one where the treasure location remains unchanged, so the agent should continue to assume the distribution represented by its internal model \hat{T} . These two cases can be combined in a weighted sum to determine a new internal distribution T'. The probability for every state t in this new distribution can be computed as

$$P(\hat{T}'=t) = P(change)\frac{1}{n} + (1 - P(change)) \cdot P(\hat{T}=t). \tag{8}$$

To model the uncertainty, this formula is applied to the agent's internal model each turn after it has completed its action. Note, that this leaves the ordering of probabilities from the most likely to the least likely event intact, unless the probability of change is 1.0. Therefore, the single agent behaviour with modelled uncertainty performs just as well as the agent without for a non-changing treasure location. But, applying the above uncertainty model to a single agent in a world where the treasure location does change, increases its performance from 0.148 (for the agent without uncertainty) to 0.180.

The performance increases because by modelling uncertainty, the agent retains some information about the order in which it explored the previous locations in its internal model. The location that was visited first and found empty subsequently had uncertainty applied to it nine times, once the agent cleared the last, tenth location. It therefore has the largest probability to contain the treasure, and will be the first location to be visited again. This actually reflects the fact that this location is most likely to contain the treasure, since it is unclear when the treasure changed location.

This also shows why modelling the uncertainty works better than simply resetting the probabilities after all locations were visited and found empty. This would reset the internal model and prevent the agent from having to use random search, but it would not preserve the information about the ordering of the previous search, which could be used to the agent's advantage.

Uncertainty and Social Bayesian Update

In this section, we examine a population where all agents model the change uncertainty and also perform the social Bayesian Update. As a result, the agent's performance drops to 0.1, which is equivalent to chance. Closer analysis shows that the whole agent population is always exploring the same location, and the 0.1 average performance is simply the result of the treasure randomly moving to this location from time to time. The agent population here is subject to an information cascade that synchronizes the whole population. But compared to the all-social agent population with internal certainty, the agents cannot reliably check that a certain location is wrong, so after the initial agent breaks the symmetry, the repeated exposure to other agent's social signals will always override their own internal uncertain beliefs. So, while the Social Bayesian Update is beneficial for agents in some cases, it turns out that it can be harmful, specifically when combined with a more accurate model of uncertainty. This is similar to how bounded rationality, i.e. the inability to internally represent certainty, facilitates convergence in social Bayesian network learning (Acemoglu et al., 2011). The difference here is that the lack of internal certainty is not caused by a limitation of the agent, such as cost of internal representation, but motivated by an increase in performance resulting from a more exact modelling of the noise present in the environment.

Partial Observability

One way to reduce the probability for convergence is the reduction of network connectivity (Gale and Kariv, 2003). Currently, the agents live in a neighbourhood of a fully connected graph, being able to observe all other agents. The next simulation has changing treasure locations and an all social, internally uncertain agent population. Unlike the previous models, only a fraction of the other agents' actions can be observed. Every time an agent takes an action, each other agent has a probability of p_o to observe this action and update its internal model. Whether an agent can observe a specific action is determined for each observing agent separately. This creates several simulations interpolating between two previously studied cases. If $p_o = 0$, then the model would be identical to the non-social agent simulation, and if $p_o = 1$, then it would be identical to one in which all agents could observe each other, which leads to a feedback loop and very bad performance ratios.

Changing Observation Probability for all Agents Varying the parameter p_o for all agents results in performance ratios as depicted in Fig. 2. As expected, the extremal points are characteristically similar in performance to the nonsocial and all-social models. In the case where no agents observe each other, the agents find the treasure on average 0.18 times per turn. The performance ratio increases as the chance to observe other agents increases, up to $\approx 30~\%$ observation probability, where all agents have a performance ratio of ≈ 0.32 . Increasing the observation probability further however, lowers the performance down to approximately 0.1 at an observation probability of 50 % and above.

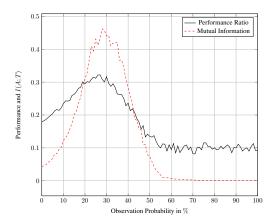


Figure 2: Average performance of an agent population, and the mutual information between its actions and the treasure location, depending on the probability to observe the actions of other agents.

The second graph (dotted red) in Fig. 2 is the mutual information between the agent's actions A, and the treasure location T. We see that I(A;T) has the same value as for a non-social agent when the observation probability is zero, it then rises to a peak of ≈ 0.45 bits for an observation probability of 30 %. The mutual information then decreases for larger observation chances, down to zero mutual information for values above 60 %.

Changing Observation Probability for one Agent If the observation probability is understood as the result of an agent's effort invested in observing others, then it could be treated as a behavioural parameter that the agent, or at least the process that governs the adaptation of agents, could control. This could be realized by deliberately degrading the agent's sensors to save resources in case of an adaptation process on the agent's population, or by simply discarding some of the sensor input at random if this is realized as an agent strategy. In this context, it would make sense to ask if an individual agent could perform better than the rest of the population by unilaterally changing its probability to observe others.

Given that the actions of the remaining population provide a high degree of mutual information, it might be useful to obtain more of this information than others do. On the other hand, there were indications that taking in too much information from other agents might override the information from the non-agent environment, and thereby degrade the agent's performance. So deliberately lowering the social information intake might also improve the agent's performance compared to the rest of the population.

In the next simulation we will look at one agent that can change its observation probability independently from the rest of the population. The observation probability for an

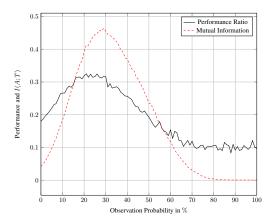


Figure 3: Performance of a single agent, and the mutual information between this agent's actions and the treasure location, depending on the probability to observe the other agents in the population. All other agents observe each other with a probability of 30%.

agent determines how well it can see others, not how well it can be seen. That means that whenever this agent could observe another agent's action, its own observation probability would be used to determine whether this agent could actually sense what action the other agent took.

All other agents in the simulation have a fixed observation probability of 30 %, since this was the value that led to the best performance for the overall population, and also encoded the most information.

In Fig. 3 we see the resulting performance ratio and mutual information I(A;T) for varying p_o for the one agent that can change its observation probability. Overall, the graph looks very similar to the previous graph in Fig. 2 where all agents could change their observation probability. The performance for one agent is still optimal at $\approx 30\%$. Scaling down the observation probability to zero obviously leads to the same performance as the non-social agent. Increasing observation probability further also results in lowering the performance to approximately 0.1.

This is particularly interesting because, for this specific simulation, it creates something akin to a game theoretic equilibrium at the 30 % point. All other factors being equal, even if all agents could change their own observation probability at will, none of them could change it away from 30 % without also decreasing their performance.

Relevant Information Analysis

So far, we have computed the mutual information between the agent's actions and the environment as a measure of how much information their collective actions provide about the state of the environment to an observer. We will now compare this mutual information to the actual relevant information for different performance levels. This will demonstrate that higher observation probabilities are characterized by processing information that is not necessary, indicating the perpetuation of false beliefs in the agent population.

 $\mathbf{RI}(\mathbf{u})$ for the Treasure Hunter Model The relevant information for the treasure hunter model is determined by the distribution of the treasure, encoded in T, and a specific agent's action distribution, encoded in A. Both random variables are defined over the same alphabet, which corresponds to all possible locations in the world.

As relevant information is a property of the environment, and not of a specific agent, it therefore considers all possible strategies p(a|t), regardless of how any specific agent would acquire the information needed to actually implement this strategy. To determine the value for RI(u) we have to answer the question, which joint distribution of A and T having at least a performance level of u has the lowest mutual information?

For our specific example of a world with ten locations we can compute the relevant information function as

$$RI(u) = \log(10) + \left(u\log(u) + (1-u)\log\left(\frac{1-u}{9}\right)\right).$$

Note that this function computes the minimal mutual information for being on a specific performance level u, not for having a strategy that at least has the performance level u. However, looking at the actual function, which can be seen in Fig. 4, it becomes clear that the function is, for values of u over 0.1, strictly increasing. Therefore, the minimal mutual information for a specific performance level above 0.1 is also the actual relevant information needed to perform at least that well. The previous distinction is necessary, because in this case it is necessary to process information to have a performance level lower than 0.1. A performance of 0.1 can be achieved with a random strategy, and therefore has no relevant information. Eq.(9) reflects this, as it is zero for u = 0.1. For values of u lower than 0.1 the function in Eq.(9) computes values higher than zero, which would be the information necessary to actually perform at this level. One would have to actively avoid the treasure. But by previous definition relevant information should return the information needed to at least attain a specific level, and since random performs better, and has no relevant information, all performance levels below u = 0.1 have zero relevant information.

The data points plotted in Fig. 4 are taken from the two previous simulations, those where all agents changed their observation probability, and those where only one agent changed its observation probability and all other agents had an observation probability of 30 %. Each point is the combination of the mutual information I(A;T) and the achieved performance ratio for a specific percentage of observation probability. Different observation probabilities result in

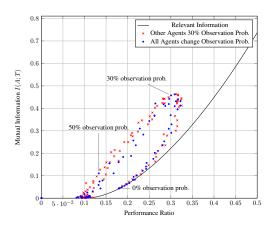


Figure 4: Relevant Information trade-off curve (black line) and points indicating the mutual information and performance for different observation probabilities.

different strategies, i.e. different conditional probabilities P(A|T).

The data points gathered here are, as expected, all above or on the RI trade-off curve. The pattern of values are very similar for both simulations. For an observation probability of 0.0 the data point is located at a performance of 0.18, and actually on the trade-off curve. As the observation probability increases, so does the performance. The strategies remain on the trade-off curve at the lower percentages of observation probability, and since the trade-off curve is strictly increasing, so does the encoded relevant information.

As the observation probability increases we see that the resulting data points leave the trade-off curve, which means the resulting strategies encode more mutual information about the environment than is necessary. The strategies resulting from further increases in observation probability are located in the upper loop where they gravitate towards a point of no mutual information and a performance of 0.1. This indicates that they also encode more information about the environment than necessary.

Comparison of the mutual information in the actual strategies to the actual relevant information illustrates how observing more and more agents leads to processed information which might not necessarily be relevant. The strategies with low observation probability are located on the relevant information trade-off curve, meaning they are efficient in the sense that they do not process non-relevant information. Those strategies which are subject to the information cascade on the other hand do display a lot of information about the environment in their actions which is non-relevant. At the same time, as seen here, their performance diminishes as well. Fortunately for the agent population, the point where agents display the most relevant information about the environment is also close to the point where the agent performs best, so it would be possible for an agent population, which

could adjust their observation probability, to stabilize at a point which benefits all agents the most.

Conclusion

Our results indicate that a noisy internal representation seems to be an important factor for the convergence of information cascades, specifically those where the agents perpetuate information that leads to wrong internal beliefs, since the agent cannot, with certainty, reject certain social information. In general, the problem arises in scenarios where signals gained from other agents overpower the agent's private observations and are not as independently generated as the naive Bayesian update models it. On the other hand, the information from other agents is also helpful, and can improve an agent's performance in our model. The interesting observation here is that both things can be influenced by how many other agents an agent randomly observes. Too little, and the agent loses the social information, too much, and the agent population will converge, but possibly on the wrong belief.

Our relevant information analysis also shows how the quality of the information suffers when more and more agents observe each other. For very low observation probabilities, all the information processed is relevant and agents only display relevant information in their actions. When the agents observe more, their performance gets better still, but we see that they start to pass on information that is incorrect and perpetuate it, sometimes leading to false convergences. As the "good" relevant information is still improving, this unnecessary information seems acceptable, but if we increase the observation chance even further, then we see that the performance suffers and the information provided by the agents is mostly wrong.

In our model however, there exists a point where agents both perform optimally and provide the most information, so a population of agents could adapt to a strategy where they discard a certain percentage of their observations, and perform well. In this case, the agents would basically determine the observation network of the model themselves. The exact parameter of how many of one's observations one should discard is, of course, model dependent. For example, if the number of agents increases, then it likely takes more observations for total convergence, but a lower observation probability could be sufficient to provide enough social information to overpower the agent's internal beliefs. This is interesting if this is seen as a model for fads and fashions. If an agent, adapted to a population with a specific degree of connectivity, adapts to discard a certain percentage of social information, and is then transplanted to another population, with different parameters, it might become much more susceptible to false self-perpetuating beliefs. The same is true for a population of agents that manages to change their environment in a way that radically changes how much they can observe others.

Acknowledgements

This research was supported by the European Commission as part of the CORBYS (Cognitive Control Framework for Robotic Systems) project under contract FP7 ICT-270219. The views expressed in this paper are those of the authors, and not necessarily those of the consortium.

References

- Acemoglu, D., Dahleh, M., Lobel, I., and Ozdaglar, A. (2011). Bayesian learning in social networks. The Review of Economic Studies, 78(4):1201–1236.
- Banerjee, A. (1992). A simple model of herd behavior. *The Quarterly Journal of Economics*, 107(3):797–817.
- Bikhchandani, S., Hirshleifer, D., and Welch, I. (1992). A theory of fads, fashion, custom, and cultural change as informational cascades. *Journal of Political Economy*, pages 992–1026.
- Easley, D. and Kleinberg, J. (2010). *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press.
- Gale, D. and Kariv, S. (2003). Bayesian learning in social networks. Games and Economic Behavior, 45(2):329–346.
- Kao, A. B. and Couzin, I. D. (2014). Decision accuracy in complex environments is often maximized by small group sizes. Proceedings of the Royal Society B: Biological Sciences, 281(1784):20133305.
- Polani, D. (2009). Information: Currency of life? *HFSP journal*, 3(5):307–316.
- Polani, D., Martinetz, T., and Kim, J. T. (2001). An information-theoretic approach for the quantification of relevance. In ECAL '01: Proceedings of the 6th European Conference on Advances in Artificial Life, pages 704–713, London, UK. Springer-Verlag.
- Polani, D., Nehaniv, C. L., Martinetz, T., and Kim, J. T. (2006). Relevant information in optimized persistence vs. progeny strategies. In Artificial Life X: Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems, pages 337–343. The MIT Press (Bradford Books).
- Salge, C. and Polani, D. (2011). Digested information as an information theoretic motivation for social interaction. *Journal of Artificial Societies and Social Simulation*, 14(1):5.
- Shannon, C. E. (1948). A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423.
- Surowiecki, J. (2005). The Wisdom of Crowds. Anchor.
- Wang, X. R., Miller, J. M., Lizier, J. T., Prokopenko, M., and Rossi, L. F. (2012). Quantifying and tracing information cascades in swarms. *PloS one*, 7(7):e40084.

The Effect of Network Structure on the Spatial Coevolutionary GA

Morgan McLaughlin and Mark Wineberg

University of Guelph

Abstract

Using the simple one-max problem we will show the range of effects spatial networks have on spatial coevolutionary Genetic Algorithms (GAs). Non-coevolutionary spatial GAs have had their spatial reproductive structures tested to show that different structures can result in varying performance; Coevolutionary variants however have not. In this extended abstract we show that varying types of spatial structures can impact the coevolutionary GA differently than the standard GA.

Introduction

Spatial and Coevolutionary GAs are two fields that have received a fair amount of research, however their behaviour when combined is not well understood. Some preliminary work has been done showing that combining spatial and coevolutionary GAs can both greatly improve the GA's performance and stabilize problematic coevolutionary behaviour such as fitness dissociation between populations(Hillis, 1991; Pagie and Hogweg, 1997; Weigand and Sarma, 2004; Mitchell, et. al., 2006). Looking at the behaviour of specific spatial structures on Spatial GA's has been a topic of interest for many researchers (Sarma and De Jong, 1996; Bryden et al, 2006); however, none of the researchers we surveyed had tried any spatial structure on a coevolutionary GA other than a simple grid. We aim to look at the effects of a number of spatial structures and view their effects on the coevolutionary GA, including whether they induce the same performance as they do on a standard GA.

Background

The Spatial GA uses graphs structures that can be found in any introductory textbook on graph theory (such as Harry, 1969). The GA uses *undirected* edges, meaning that each connection implies that the both nodes are aware of each other. We will also discuss: the *degree* of a graph, the average degree (number of connections) of all nodes in that graph; the *path length* between nodes, the number of connections necessary to get from one node to another; and the *diameter*, the maximum shortest path length between any two nodes.

A GA with a spatial reproductive network is one that only allows an individual to reproduce with nodes within its direct neighbourhood. This, in effect, limits the amount of genetic flow which can occur through the population (Sarma and De Jong, 1996). A standard GA can be considered a spatial GA with a complete network, wherein every node is connected to every other node.

Methodology

GA Setup

We use the simple one-max problem to quickly and succinctly show that the impact of spatial structures on the standard GA is different than the same problem on the coevolutionary GA using a standard easy to understand function. The fitness of a chromosome solving one-max is simply equal to the number of ones it contains; this implies the maximum fitness is the chromosome of all ones.

The coevolutionary GA is setup so that each population is placed on identical reproductive networks and for the evaluative network each node from one population is paired with a node that has the same reproductive location in the other population. The fitness of each coevolutionary individual is set equal to the combined one-max score of itself and its partner, i.e. we are examining a cooperative coevolutionary system. Our standard GA uses a chromosome length of 60 and the coevolutionary GA uses a chromosome length of 30 for each population. These values are comparable as the coevolutionary GA is mapping two chromosomes together. We are using an elite size of 2 (the members with the two largest fitness values are reproduced using local elitism), a 0.8 probability of crossing over using uniform crossover with a parameter of 0.3 and a mutation rate of 2/L where L is the chromosome length for both GAs. The GA is run until one-max is found up to 5000 generations when it is cut off; any GA that reaches 5000 generations is reported as reaching that number. We run 50 repetitions for each spatial structure on each GA configuration.

Spatial Structures

We will be comparing the performance of one-max on 8 different spatial structures for both GA types. These structures will include 2 grid type, 3 ring type and 2 random type graphs. Figure 1 shows the first of the grid type $Grid\ 4$ (so named here as it has a degree of 4), which is the structure most commonly seen in spatial coevolutionary papers. Figure 2 is $Grid\ 8$, connecting 8 nodes, the same 4 that Grid 4 is connected to and additionally the 4 nodes on the diagonal. 'Ring 2' is a standard ring, $Ring\ 4$ (Figure 3) is a ring where each node is also connected to its neighbor's neighbor, implying a degree of 4. $Ring\ 8$, is the same as Ring 4 but connected to the 4 nodes on each side. $Rand\ 4$ and $Rand\ 8$ which are random graphs, are reconstructed for each GA run, with an average degree of 4 and 8 respectively; and finally the complete graph which simulates the standard GA.

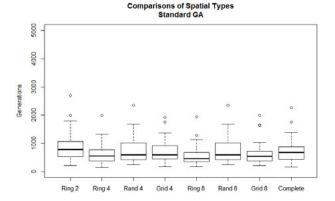


Figure 4: Spatial Structures on the Standard GA

The node's degree provides the minimum time for genetic information to travel between individual nodes, while the diameter measures the minimum time for genetic information to travel everywhere in the population. Each structure has a degree equal to the number in its name and the following diameters: 9 for Grid 4, 4 for Grid 8, 50 for Ring 2, 25 for Ring 4, 13 for Ring 8, and 1 for Complete. For Rand 4 & 8 the diameter varies based on the run and can even be infinite if a disconnected graph is formed.

Results

The results for the experiments done on the standard GA can be seen in Figure 4, while the coevolutionary results are shown in Figure 5. We can quickly see that the worst result received was for the complete coevolutionary GA; it was the only GA to report any failed solutions within the 5000 generation limit and is much worse than every other GA tested. The spatial structures at each connection level have been tested using the Wilcoxon rank-sum test with a Holms-Bonferroni correction and 99% confidence level and they all show a statistically significant difference in performance. The diameters of the graphs appear to have no, or at least a very limited, effect. Notice that while the random graphs have a larger variance than the other graphs, it is nowhere near as large as we might expect if diameter was playing an important role. It is clear that for the one max problem on the coevolutionary GA the graphs with the lowest degree provide the best performance.

In stark contrast to the coevolutionary results, we see that the spatial structures have barely any effect on the standard GA for this one max problem; no structure is statistically significantly better than the rest. Somewhat surprisingly, we found that the GA with the best performance on the one-max problem is the Coevolutionary GA with a Ring 2 structure, the next best performing are the 3 graphs with 4 connections on the Coevolutionary GA; they have better performance than the Standard GA when compared with a 99% confidence level. Finally, it is clear that spatial locality does help stabilize the coevolutionary system, at least for the one-max problem, as evidenced by the poor performance when using the complete graph in Figure 5.

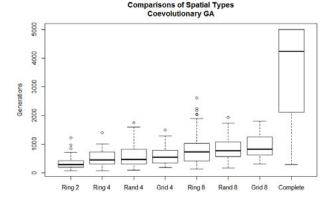


Figure 5: Spatial Structures on the Coevolutionary GA

Conclusion

It is clear that the impact of spatial structures on the coevolutionary GA has different effects than on the standard GA. We have only shown the difference using the simple onemax problem, but it is clear that getting such a variation in performance on such a simple problem will likely imply similar performance differences on more complex problems; though more experimentation would be required to determine which structure is best. We have shown that, at least for the one max problem, connectivity is more important than diameter; i.e. local dissemination is more important than global dissemination. We hypothesize that this is due to a 'locking in' effect, wherein good genes are able to stay near each other in both populations and synergistically work towards the optimum. More evidence supporting this hypothesis through looking at elitism can be seen in McLaughlin and Wineberg, 2014. We believe that this should hold for more complicated problems run on a coevolutionary GA, and consequently, a more detailed study is required.

References

Bryden, K. M., et al. (2006). Graph-based evolutionary algorithms. *Evolutionary Computation*, IEEE Transactions 10.5: 550-567.
Harary, F. (1969). *Graph Theory*, Addison-Wesley, Reading, MA.
Hillis, W. D. (1990). Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*: 42.1: 228-234.
McLaughlin, M. and Wineberg, M. (2014). The Effects of Elitism on Spatial Coevolutionary GAs. To appear in *Artificial Life 14*.
Mitchell, M., Thomure, M. and Williams, N. (2006). The role of space in the success of coevolutionary learning. *Artificial Life X*, 118-124.
Pagie, L. and Hogeweg, P. (1997). Evolutionary consequences of coevolving targets. *Evolutionary Computation* 5(4): 401-418.
Sarma, J. and De Jong, K. (1996). An analysis of the effects of neighborhood size and shape on local selection algorithms. *Parallel Problem Solving From Nature IV*, 236-244.
Wiegand, R. P. and Sarma, J. (2004). Spatial embedding and loss of

gradient in cooperative coevolutionary algorithms. Parallel Problem

Solving From Nature VIII, 912-921.

Evolution as a Random Walk on a High-Dimensional Manifold Defined by Physical Law: Implications for Open-Ended Artificial Life

Andrew L. Nelson¹ and Brenae L. Bailey²

¹Androtics LLC, Tucson, Arizona 85733, USA

²Department of Mathematics, University of Arizona, Tucson, Arizona 85721, USA <u>alnelson@ieee.org</u>

Extended Abstract

This short article presents a discussion of the underlying conditions under which natural evolution of life occurs and how these natural conditions may be extremely difficult to implement in artificial life (ALife) systems. In particular, the Darwinian concept of adaptation via natural selection may not have a complete or functional macro-level description that could be used to build any evolutionary environment that is defined at the agent-environment interaction level.

Here we are specifically addressing open-ended evolution (Ruiz-Mirazo et al., 2004) in artificial systems (Standish, 2003; Nolfi, 2012; Mouret and Doncieux, 2012). One of the goals of ALife is to generate systems capable of sustained evolution of life-like complexity. Such systems, although artificial, could then be considered to produce real life of a kind (Pattee, 1987; Ray, 1993), as opposed to being just simulations of life.

For evolutionary computing applications aimed at solving particular problems, a well-defined goal that is separate from survival in and of itself can be formalized into a selection criterion and used to evolve solutions (Oduguwa et al., 2005). However, in the case of open-ended evolution, many researchers now accept that agent-level definitions of fitness are unsuitable to drive differential selection and replication (Lynch, 2007; McShea, 1991; Lehman and Stanley, 2011). This includes even the most unbiased and high-level implicit fitness criteria in which replication is seemingly made to be a direct result of agent interaction with the environment (see for example Yaeger, 1994). Interestingly, concerns about the adequacy of neo-Darwinian and Darwinian theory to fully describe the evolution of life have come from several ALife researchers, often after attempting to implement evolving systems (Mitchell and Forrest, 1994; Lehman and Stanley, 2011; Watson, 2012; Nolfi, 2012). Ray, for example, indicates that there is something "oddly self-referential" about evolution (Ray, 1993). Dawkins describes how his views of natural biological evolution changed after playing around with ALife simulations (Dawkins, 2003).

An underlying tenet of science is that all observable natural phenomena result from a fundamental set of physical laws and that physical law is essentially unchanging (Feynman, 1967; Zilsel et al., 2003). Such a set of laws, although not yet fully elucidated by physicists, is presumed to exist. If this were not the case, some fundamental cornerstones of science such as

repeatability of experiments, as well as a host of epistemological underpinnings, would not hold. A consequence of the existence of such a set of elemental physical laws is that fundamental driving forces producing change in natural evolution result from or reflect the topology of a *static* space defined solely by unchanging physical law (Ray, 1993). In this sense (and noting that physics is thought to have an intrinsic stochastic aspect), evolution can be described as a random walk on a static manifold, one of extremely high dimensionality. The only fundamental non-random "force" driving change in nature is imparted by the underlying topology of this static extremely low-level and high-dimensional landscape. Furthermore, this low-level view of the universe is not mediated by a replication cycle per se.

The discussion above implies that a system defined only in terms of a suitable set of elemental rules might in theory support open-ended evolution, and that our natural universe is an example of such a system. This raises the possibility that high-level representations (including the differential survival and replication paradigm upon which Darwinian evolution is based), while describing evolution sufficiently to generate simulations, might not fully functionally specify evolution to the degree needed to generate artificial realizations of evolution. (See Pattee (1987) for a discussion of the distinction between simulation and realization.) Below, we loosely summarize an argument that implies that high-level descriptions of complex systems are likely to be functionally incomplete.

When complex systems with a high level of granularity are converted to lower levels of resolution, information is usually lost, even if overall patterns are seemingly more evident (Katsoulakis and Trashorras, 2006). Hence, if the behavior of a complex system is fully described (but not over-specified) at one level, it is in fact not likely to be fully described at a reduced level of resolution. The implication is that macrolevel traits in biological systems, being essentially extremely low-resolution views of matter/energy configurations, do not contain sufficient information to fully predict replication efficiency distributions (as generalizations of adaptive fitness landscapes (Wright, 1932) might have suggested).

Relating variation in macro-level traits to replication efficiency would then not fully define the underlying forces driving evolution, not even in theory. In this case, the

paradigm of natural selection would still be useful in a retrospective sense for summarizing some high-level relationships between observed phenotype and reproductive efficiency (Kauffman, 1993), but would not be sufficient to functionally define or drive open-ended evolution of such complexity (Lynch, 2007; McShea, 1991).

We believe this view has relevance to the long-term success of ALife. A guiding approach employed in much of ALife is that of setting up an environment in which agents defined at the macro level compete to survive and replicate. This approach to producing systems capable of supporting openended complexity may be fundamentally flawed, even when implemented without overt bias and with asynchronous local reproduction and careful attention to definitions of fitness as highlighted in Lichocki et al. (2012).

Although obtaining an explicit description of a given complex phenomenon solely in terms of elemental physical law may be intractable, we maintain that it is not theoretically impossible. Further, this may be the only level at which complex biological phenomena are completely causally described.

A system defined by elemental physical laws is clearly sufficient to produce open-ended evolution, as this describes life in our own universe. However, in order to generate an artificial system capable of the open-ended evolution of complex agents, it may be not merely sufficient but necessary to define environments in terms of elemental rules (Ray, 1993). Agents in such a system must either be constructed using only these rules, or perhaps arise through abiogenesis, as natural life did. If complex self-replicators and their environment are constructed from a single set of consistent rules, the system could be considered to contain *endogenous* ALife. Currently such systems remain beyond the state of the art (Nelson, 2013), but recent work in artificial chemistry and soft ALife have made considerable advances in this direction (Joachimczak et al., 2012; Fontana, 2010).

Many questions remain. For example, at what level must in silico ALife environments be specified? Can any system defined at the macro/agent level be considered to be free of implicit fitness functions? Does the increase in complexity observed in, e.g., vertebrate evolution represent a general aspect of possible life, or is it just an artifact of life on Earth?

To summarize, in this short paper we have argued that macro-level concepts of natural selection cannot be used to define systems capable of supporting the open-ended evolution of complex life-like self-replicators. Further, the only fully explanatory driving force behind the evolution of natural life is imparted by the topology of a static low-level landscape defined by unchanging physical law. Higher-level descriptions that include differential survival explicitly linked to replication cycles are only adequate to generate simulations of evolution, not realizations of evolution in which complex agents actually arise.

References

- Dawkins, R. (2003). The evolution of evolvability. In *On Growth, Form and Computers*, pages 239–255.
- Feynman, R. (1967). The character of physical law. Vol. 66. MIT Press.

- Fontana, A. (2010). Devo co-evolution of shape and metabolism for an artificial organ. In *Artificial Life XII*, pages 19–23.
- Joachimczak, M., Kowaliw, T., Doursat, R., and Wrobel, B. (2012). Brainless bodies: controlling the development and behavior of multicellular animats by gene regulation and diffusive signals. In *Artificial Life 13*, pages 349–356.
- Katsoulakis, M. A. and Trashorras, J. (2006). Information loss in coarse-graining of stochastic particle dynamics. *Journal of Statistical Physics*, 122(1):115-135.
- Kauffman, S. A. (1993). The Origins of Order: Self-Organization and Selection in Evolution. Oxford University Press.
- Lehman, J. and Stanley, K. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19.2 (2011):189-223.
- Lichocki, P., Keller, L., and Floreano, D. (2012). Differences in the concept of fitness between artificial evolution and natural selection. In *Artificial Life 13*, pages 530-531.
- Lynch, M. (2007). The frailty of adaptive hypotheses for the origins of organismal complexity. PNAS, 104:8597–8604.
- McShea, D. W. (1991). Complexity and evolution: what everybody knows. *Biology and Philosophy*, 6(3):303–324.
- Mitchell, M. and Forrest, S. (1994). Genetic algorithms and artificial life. *Artificial Life*, 1(3):267–289.
- Mouret, J-B. and Doncieux, S. (2012). Encouraging behavioral diversity in evolutionary robotics: an empirical study. *Evolutionary Computation*, 20(1):91–133.
- Nelson, A. L. (2013). Artificial life and machine consciousness. In 2013 AAAI Fall Symposium Series: AAAI Technical Report FS-13-02, pages 52–57.
- Nolfi, S. (2012). Co-evolving predator and prey robots. *Adaptive Behavior*, 20(1):10–15.
- Oduguwa, V., Tiwari, A., and Roy, R. (2005). Evolutionary computing in manufacturing industry: an overview of recent applications. *Applied Soft Computing*, 5(3):281–299.
- Pattee, H. H. (1987). Simulations, realizations, and theories of life. In Artificial Life, pages 63-78.
- Ray, T. S. (1993). An evolutionary approach to synthetic biology: Zen and the art of creating life. Artificial Life, 1(1):179–209.
- Ruiz-Mirazo, K., Peretó, J., and Moreno, A. (2004). A universal definition of life: autonomy and open-ended evolution. *Origins* of Life and Evolution of the Biosphere, 34(3):323–346.
- Standish, R. K. (2003). Open-ended artificial evolution. *International Journal of Computational Intelligence and Applications*, 3(2):167–175.
- Watson, R. A. (2012). Is evolution by natural selection the algorithm of biological evolution? In *Artificial Life 13*, pages 121–128.
- Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding and selection in evolution. *Proceedings of the Sixth International Congress on Genetics*, 1(6):356–366.
- Yaeger, L. (1994). Computational genetics, physiology, metabolism, neural systems, learning, vision, and behavior or PolyWorld: life in a new context. In *Artificial Life III*, pages 263-298.
- Zilsel, E., Raven, D., Krohn, W., and Cohen, R. S. (2003). The genesis of the concept of physical law. In *The Social Origins of Modern Science*, Boston Studies in the Philosophy of Science 200:96-122, Springer Netherlands.

Close returns plots for detecting a chaotic source in an interaction network

Haifa Rabai¹, Rodolphe Charrier¹ and Cyrille Bertelle¹

¹Université du Havre, Le Havre, France 76600

Abstract

We are interested in studying the spread of chaos in an interaction network modeled by a Coupled Map Network (CMN). This graph is formed by nodes characterized by a measurable state variable that may exhibit chaotic time series. The interaction between the nodes may propagate their states in the network leading through a coupling process to some synchronization phenomenon which is known as nonlinear oscillator synchronization.

The interaction network that we aim to study contains initially only one chaotic node that is responsible of the spread of chaos.

Our goal consists then to study how to identify the node which is the source of the spread of chaos in an interaction network and how to detect the set of nodes becoming disturbed by the propagation of the chaotic node state.

In this paper, we seek some appropriate measures to quantify the dynamic complexity of the nodes in order to identify the group of chaotic nodes as well as the source of the spread of chaos in the graph. We show by some simulations on random graphs that the Shannon entropy calculated on the close returns plots is an appropriate measure to detect chaotic series from a node. The extension of close returns plots to joint recurrence plots enables to identify the source of the spread of the disturbance in the network.

Introduction

The study of dynamical systems interacting in a lattice has been an active field for the last years. Recently, the attention has shifted towards more general networks of coupled maps more specifically the Coupled Map Networks (Shibata and Kaneko, 2003). Such systems are composed of many interacting nonlinear elements coupled in a way that leads them to share information about each other's state. This coupling process can propagate the node states in the network leading to a synchronization phenomenon which is defined as a correlation among time series.

Our research work is part of this context. We are interested in studying the spread of one chaotic state in an interaction network whose structure is unknown. Each node of this interaction network has a dynamic measurable state variable that may exhibit potentially chaotic time series. The arcs of this graph define the interaction between nodes that is modeled by coupling in the Coupled Map Network.

We assume that at a given time t, one node can have a chaotic internal state that can spread in the network through interaction leading to disturbing the behavior of other nodes that may become chaotic in their turn.

Many interesting issues arose from this phenomenon: How can we detect the set of nodes impacted by the spread of chaos? And how can we identify the node which is the source of the disturbance in the interaction network?

In order to answer these questions, we propose to use tools for nonlinear time series analysis more specifically the Shannon entropy computed on the close returns plots to measure the dynamic complexity of the nodes. However, this measure is insufficient to identify the source of the disturbance in the network. We propose then to investigate the recurrences of the node time series by computing the mean conditional probability of recurrence, that was proposed by Kurths et al. (Romano et al., 2007), to identify the origin of the spread of chaos in the graph. This measure had been used to detect the coupling direction for 3 interacting chaotic systems. In our case, we compute this measure for more than 3 systems with only one chaos generator which is the source of the spread of chaos.

Studying the behavior of the interacting elements is partly motivated by results showing that the structure of a network can impact the dynamical properties of the system (Pereira et al., 2013).

Moreover, this work is part of a real life problem which consists in modeling the spread of panic in pedestrian crowds during emergency situations. Assuming that the phenomenon of the spread of panic is faster than the pedestrian mobility which is a reaction to this propagation, we can extract the configuration of the crowd and thus obtain an instantaneous image of it. This static configuration corresponds to an interaction network where the nodes represent the pedestrians and the arcs are the interaction between them.

With regard to panic, we can associate the panicked state of the pedestrians to a node state variable having chaotic time series. This idea is partly justified by the studies made by some psychiatrists and neurologists who were interested in the internal state of the individuals. They wanted to know the nature of the dynamics of the breaths and heart rate of some patients suffering from panic disorder. They compared them to those of healthy persons and showed that they had a greater entropy than the breaths of healthy individuals. Therefore, they concluded to the presence of chaotic disturbances in the breath data series due to the panic effect (Caldirola et al., 2004).

Studying the spread of chaotic states in an interaction network enables to prepare experimental data to be used to study the spread of panic. In fact, thanks to the development of mobile devices, we may anticipate the possibility of obtaining real data by recovering physiological data in panic situations.

The paper is organized as follows. First, we give the model for our coupled map networks. We also define chaotic time series and explain how to generate them. After that, we present the first measure selected to identify the chaotic nodes which is the Shannon entropy computed on the close returns plots. Besides, we give an overview of the mean conditional probabilities of recurrence computed on the joint recurrence plots that we intend to use to identify the node which is the source of the spread of chaos in the interaction network. The last section presents our simulations and the results obtained for each measure. Finally, we summarize the main ideas presented in this paper and present some perspectives.

Interaction network

Coupled Map Network (CMN) (Koiller and Young, 2010) is a dynamic system with a discrete time step and continuous states space. The dynamic nodes of the model are distributed in space and interact with each other by coupling. As a result of interacting dynamic elements, we can observe in the CMN an interesting synchronized behavior (Pecora et al., 1997).

To obtain a chaotic state variable, we need to use a chaos generator which is in our case the logistic map (Phatak and Rao, 1995). This nonlinear application enables to define the state variable of the nodes thanks to its control parameter and to couple many state variables.

A node's state x generated by this nonlinear application is given by:

$$x^{t+1} = f(x^t, a) = 4 a x^t (1 - x^t)$$
 (1)

where a is the control parameter of this application and t is the time step. According to the bifurcation diagram (cf. figure 1, when a is equal to 1, we have a "chaotic state", that is a chaotic time series which is characterized by a positive Lyapunov exponent (Rosenstein et al., 1993). On the other hand, when this parameter is inferior to about 0.89, we only

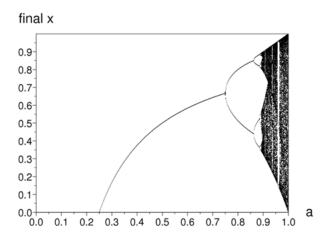


Figure 1: Bifurcation diagram of the logistic map.

have periodic or fixed point behaviors. The dynamics for the nodes are given by the following global equation:

$$X^{t+1} = ((1 - \epsilon)I + \epsilon CG)F(X^t)$$
 (2)

where X is the vector of dynamical variables of all nodes, ϵ is the coupling parameter, it also may be a matrix if each node has its own interaction strength. I is the identity matrix, $F(X) = [f(x_1^t), f(x_2^t), ..., f(x_n^t)]^T$ the nonlinear function applied to each node state and G the adjacency matrix with elements taking value 1 if node X_l interacts with node X_k and 0 otherwise. Thus, depending on the interactions between the nodes, this matrix looks like for example:

$$G = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & \ddots & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

C is a diagonal matrix that represents the coupling between one node and the other nodes. It should look like the following:

$$C = \begin{pmatrix} 1/N_1 & 0 & 0 & 0\\ 0 & 1/N_2 & 0 & 0\\ 0 & 0 & \ddots & 0\\ 0 & 0 & 0 & 1/N_n \end{pmatrix}$$

 $N_1,\ N_2,\ ...,\ N_n$ are the number of agents interacting with each node. (ϵ C G) corresponds to our interaction network. This coupling gives rise to the spread of chaos in the interaction network. In order to identify the set of nodes impacted by this propagation, we need a measure to detect the chaotic time series. One may think of using the Lyapunov to achieve this goal. However, it is difficult to compute the

Lyapunov exponent on any kind of interaction network nor on experimental data (Wolf et al., 1985). So, we will use an other measure more easier to compute and more adapted to all kinds of situations to detect chaotic times series. This measure is the Shannon entropy computed on the close returns plots. We also propose to use the extension of the close returns plots to joint recurrence plots to compute the mean conditional probabilities of recurrence that enable to detect the coupling directions between the nodes by following the work of Kurths et al. In the following section, we will present these measures.

Detecting chaotic nodes

In order to characterize the dynamical properties of the nodes time series, we propose to use the recurrence quantification analysis and more precisely the Shannon entropy computed from the close returns plots (Mindlin and Gilmore, 1992).

Close returns plot is an array matrix $N \times N$ that is build as follows. Let x_k be the time series of a given node X_k and N is the number of samples needed to build the close returns plots. This number must be sufficiently large, generally it is superior to 500.

The principle consists in comparing this time series at each time step i to itself with a delay j. It is noted that for the sake of clarity, the time step will be denoted i to fit the matrix notation below.

$$R_{X_k}(i,j) = \theta(\left\|x_k^i - x_k^{i+j}\right\| - \delta_k) \tag{3}$$

where $\theta(x_i)$ is the Heaviside function.

If the difference is less than a certain threshold δ_k , they are considered as recurrent and represented by 1 in the matrix. Otherwise, they are said to be non-recurrent and 0 is placed at (i,j). Close returns plots are close to the recurrence plots that are graphical devices introduced by Eckmann, Hamphorst and Ruelle (Eckmann et al., 1987) to measure the recurrence properties of chaotic dynamics.

Many measures were proposed later by Trulla et al. (Trulla et al., 1996) to transform the graphical interpretations into statistical analysis. Among these measures was the Shannon entropy used in its first version to quantify the degree of the recurrence. It is given by:

$$S = -\sum_{n=1}^{H} P_n log(P_n) \tag{4}$$

where P_n is the probability to observe a recurrent segment with length n. H is the longest sequence of recurrent elements.

This definition of the Shannon entropy doesn't make possible to detect chaotic time series as it quantifies the recurrence of the dynamics. To address this problem, a new definition of this measure was proposed (Rabarimanantsoa et al.,

2007). In order to quantify the complexity, we should compute the Shannon entropy from segments of non-recurrent points in the recurrence plot. So, in our case, P_n is the probability to observe non-recurrent segments in a close return plot, that is the number of non-recurrent horizontal segments with length n>0 divided by the total number of non-recurrent segments.

Identifying the source of chaos

In order to detect the interactions between the chaotic nodes, we propose to use a method based on the recurrence properties and developed by Kurths et al. This method aims to infer coupling directions between dynamical systems based on their recurrence properties. It hinges on the recurrence plots and their extension to joint recurrence plots.

First of all, we need to compute a recurrence plot for each node which corresponds in our case to the close returns plots. Then, we perform a pairwise analysis. In this second step, we compute the joint recurrence plots between each couple of nodes X_l and X_k which gives the following matrix:

$$JR_{X_{l}X_{k}}(i,j) = R_{X_{k}}(i,j) \times R_{X_{l}}(i,j)$$
 (5)

The next step consists in computing the mean conditional probabilities of recurrence (MCR) that we present later in a form of a matrix where each element corresponds to a value of MCR computed between a couple of chaotic nodes.

The mean conditional probabilities of recurrence are defined as follows:

$$MCR(X_{l}|X_{k}) = \frac{1}{N} \sum_{j=1}^{N} p(\mathbf{x}_{l}^{j}|\mathbf{x}_{k}^{j}) \qquad (6)$$

$$= \frac{1}{N} \sum_{j=1}^{N} \frac{\sum_{i=1}^{N} JR_{X_{k},X_{l}}(i,j)}{\sum_{i=1}^{N} R_{X_{k}}(i,j)}$$

$$MCR(X_{k}|X_{l}) = \frac{1}{N} \sum_{j=1}^{N} p(\mathbf{x}_{k}^{j}|\mathbf{x}_{l}^{j}) \qquad (7)$$

$$= \frac{1}{N} \sum_{j=1}^{N} \frac{\sum_{i=1}^{N} JR_{X_{k},X_{l}}(i,j)}{\sum_{i=1}^{N} R_{X_{l}}(i,j)}$$

where $p(\mathbf{x}_l^{\mathbf{j}}|\mathbf{x}_k^{\mathbf{j}})$ estimates the probability that the trajectory of X_l recurs to the neighborhood of x_l^j under the condition that the trajectory of X_k recurs to the neighborhood of x_k^j . If X_k drives X_l , we have $MCR(X_l|X_k) < MCR(X_k|X_l)$ and we write $\Delta MCR(X_l|X_k) = MCR(X_l|X_k) - MCR(X_k|X_l)$. This inequality is explained by the difference of complexity between X_k and X_l . In fact, the dimension of X_l becomes larger than the dimension of X_k as it is determined by both the states of X_k and X_l . The variation of $\Delta MCR(X_l|X_k)$ determines the existing of coupling. If

the two nodes are independent we have:

$$MCR(X_{l}|X_{k}) = \frac{1}{N} \sum_{j=1}^{N} p(\mathbf{x_{l}^{j}}|\mathbf{x_{k}^{j}})$$

$$= p(\mathbf{x_{l}^{j}})$$

$$= RR_{X_{l}}$$
 (8)

where RR_{X_l} denotes the recurrence rate of the node X_l . The process of finding the source of chaos in the interaction network is described as follows.

- 1. Choose an initial node among the chaotic nodes.
- 2. Search the lowest MCR between this node and all the other chaotic nodes.
- 3. This MCR gives us the next node to be processed. Return by starting from this node to step 2.
- The previous loop ends when we come back to a node already visited: this node is considered as the source of chaos spread.

Simulations

In order to evaluate the efficiency of the measures presented previously, we simulated 30 random graphs formed by 50 nodes with only one chaotic node chosen randomly. In this paper, we present results related to only one example of interaction network.

To generate time series on each node through the CMN computational model, we have to define first the interaction network. This is done as follows. Each node can be considered in the real life as an agent having a limited perception of the environment. We represent this perception by arcs between a node and the set of nodes perceived by it. For example, if a node n1 perceives an other node n2, we create a directed edge from the first node to the second one. We obtain then a directed and dynamic graph.

To generate the time series of the chaotic node, we set the value of the parameter a to 1. On the other hand, we initialized the control parameter of the other nodes with random values between 0 and 0.89. Depending on the control parameter in this interval, we only have a fixed point or a periodic behavior. The end of this interval corresponds to the end of the cascade of period doubling phenomenon (cf. figure 1). We chose then a control parameter for the 49 nodes inferior to about 0.89 to make sure to have a non chaotic behavior for these nodes.

Figure 2 below shows an example of a coupled map network formed by 50 nodes. The source of the chaos that we aim to identify in the interaction network is colored in green. The red nodes are those that we expect to be influenced by the spread of chaos as they are directly and indirectly related to the chaotic node. We assume that this particular node is not

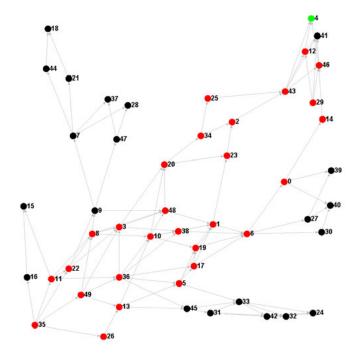


Figure 2: Coupled map network formed by 50 nodes having identifiers from 0 to 49. The green node is the source of the spread of chaos. The red nodes are those who are expected to be influenced by the propagation of chaos in the interaction network.

influenced by any other node.

When the coupling coefficient ϵ is equal to zero, no node is influenced by the chaotic node. The behavior of the nodes is then periodic or a fixed point.

When we increase the coupling parameter, we observe a modification of some node behaviors that are directly and indirectly connected to the chaotic node (cf. figure 3 and 4). We used our CMN to generate the interaction network. In the first model, we do know the source of chaos as well as the nodes that are likely to be influenced by the spread of the perturbation. On the other hand, we don't have much information about the interaction network whose structure is completely unknown. Trying to identify the source of chaos leads to the reconstitution of the CMN based on the interaction network.

Results

First, we built the close returns plots related to each node of the interaction network. To do that we used time series of 1000 successive points. We choose a threshold δ for each time series which is proportional to the mean phase space diameter in a ratio of 10^{-2} . The mean phase diameter is simply given by the difference between the min and max values of the series.

We computed the Shannon entropy related to each node.

The values obtained are either positive or equal to zero. They are reported in figure 4 The nodes that have a positive Shannon entropy, are those who have chaotic time series and thus are impacted by the spread of chaos. At this stage, the interactions between the nodes are yet unknown.

We tried several coupling parameters to see their effects on the nodes dynamic. In the following, we present 3 close returns plots and 3 time series distributions of a node impacted by the spread of chaos, computed for 3 values of ϵ : 0.1, 0.5 and 0.9.

When epsilon is equal to 0.1, we can see that the node oscillates over a short range of values corresponding to a horizontal line in blue in figure 3. This observation was confirmed by the recurrence plot in figure 3 where we only have recurrent segments represented by black points.

When we increase the value of ϵ , the behavior of the node changes and becomes disturbed. We can observe in figure 3 the difference between the 3 distributions of the time series according to the ϵ value for a given node with a basic fixed point behavior. When ϵ is equal to 0.5, the node oscillates in a completely different range of values colored in black from the interval where ϵ is equal to 0.1. We can also notice that this range widens when epsilon is equal to 0.9 and that the distribution becomes more random. This is confirmed by the two close returns plots in figure 4 (b) and (c) where we can see the emergence of non-recurrent segments which means that we have chaotic time series.

With regard to the origin of chaos spread, we computed the recurrence rates related to the nodes identified as chaotic by the Shannon entropy. The increase of the coupling parameter leads to a decrease of the nodes recurrence rate. This phenomenon is explained by the increase of the non-recurrent points due to the appearance of chaotic time series. Therefore, we don't expect to have necessarily the same inequality of the mean conditional probabilities of recurrence between the driver and the response as proposed by Kurths et al. Rather, this inequality is based on the difference between the recurrence rate of the driver and the response. Suppose X_k is the driver and X_l is the response, we may have $RR^{X_k} < RR^{X_l}$ and therefore, $MCR(X_k|X_l) < MCR(X_l|X_k)$.

In order to identify the source of chaos, we computed the mean conditional probabilities of recurrence between every couple of chaotic nodes. We report some results in the matrix below. They only concern the MCR values obtained for the interaction network presented in this paper and not for the 30 random graphs. The diagonal element of this matrix corresponds to the recurrence rate of a node X_k .

According to the inequality of the recurrence rates presented previously, a node who is less impacted by the spread of chaos, is more susceptible to have a greater recurrence rate and therefore a greater MCR.

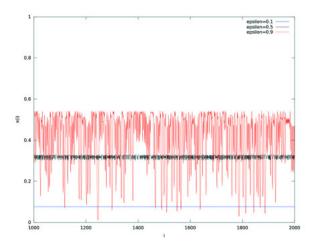


Figure 3: Distribution of time series of a node having a positive Shannon entropy during 1000 time steps for 3 values of ϵ .

	4	12	43	25	2	14	0	
4	/0.18	0.79	0.79	0.22	0.22	0.19	0.18	\
12	0.85	0.20	1	0.26	0.26	0.26	0.21	
43	0.85	1	0.20	0.26	0.26	0.26	0.21	
25	0.48	0.52	0.52	0.40	1	1	0.49	
2	0.48	0.52	0.52	1	0.40	1	0.49	
14	0.48 0.67	0.52	0.52	1	1	0.40	0.49	
0	0.67	0.69	0.69	0.80	0.80	0.80	0.66	
÷	(:	÷	÷	÷	÷	÷	÷)

Following this matrix, if we consider any node in the set of chaotic nodes, we can trace a set of paths that lead to the source of chaos. In fact, according to the inequality between the mean conditional probabilities of recurrence, the more the nodes are impacted by the spread of chaos, the less they have recurrent points in their close returns plots.

In figure 6, we can see an example of a path starting from node 48 and leading to the source of the spread of chaos: node 4. This result matches the configuration of our Coupled Map Network.

We tried to detect the arcs between the chaotic nodes based on the inequality between MCR values, we obtained a complete graph.

In order to evaluate the efficiency of the mean conditional probabilities of recurrence, we computed ΔMCR between the nodes that were identified as chaotic by the Shannon entropy and the other nodes. The results obtained showed that there's no dependence in terms of coupling between the two sets of nodes. For example, Suppose, X_l is a non chaotic node and X_k is a chaotic one, we have: $MCR(X_l|X_k) = RR_{X_l}$ which means that the nodes are independent.

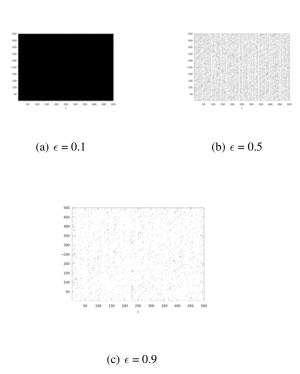


Figure 4: Close returns plots computed for 3 values of ϵ for a node having a positive Shannon entropy.

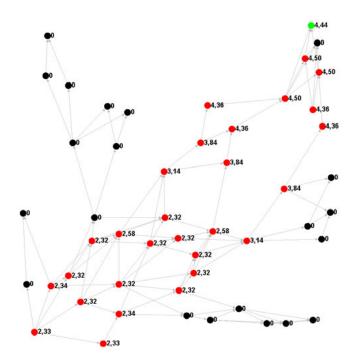


Figure 5: Chaos spread in the interaction network detected by the Shannon entropy. The nodes colored in red have a positive Shannon entropy.

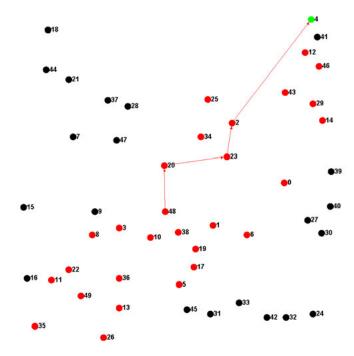


Figure 6: A path from a chaotic node to the source of the spread of chaos colored in orange.

Discussion

The use of the Shannon entropy has been successful in detecting the chaotic nodes. However, this measure is insufficient to quantify how the nodes are impacted by the spread of chaos in the network. The mean conditional probability of recurrence computed on the joint recurrence plots enables to deal with this issue. In fact, computing this measure depends partly on the recurrence rate of the nodes which indicates the effect of the spread of chaos on the nodes dynamics. Thus, the nodes who are more impacted by the spread of chaos, has a smaller recurrence rate.

For example, if we compare two nodes X_l and X_k and obtain $MCR(X_l|Xk) < MCR(X_l|Xk)$, it will mean that X_l is more impacted by the spread of chaos than X_k . Kurths et al. detected the coupling directions between 3 chaotic systems. In contrast to this work, we only have one chaotic node in the network that spreads the chaos. As a result of coupling, the recurrence rate of the nodes being impacted by the chaos decreases. This phenomenon explains why we don't have necessarily the result as Kurths et al.

The approach aiming to detect the direct coupling directions was extended by Marwan et al. (Zou et al., 2011) to infer indirect coupling. The principle of this method relies on the fact of computing the contribution of a node X_z in the mean conditional probabilities of recurrence of X_l and X_k . We tried to use this method to detect the chaotic network structure but we couldn't achieve our goal. This is explained

by the fact that all the nodes that are impacted by the spread of chaos inherit their behaviors from the same node. Therefore, two chaotic nodes that aren't directly interconnected to each other are somehow related to each other as a result of coupling with the same node.

Conclusion

In this paper, we presented our research work which consists in studying the spread of one chaotic state in an interaction network as a result of the interaction between the nodes. This interaction is modeled by a coupling process.

We used the Shannon entropy computed on the close returns plots as a measure for detecting the sub group of chaotic nodes in the graph. The chaotic influence between this set of chaotic nodes is detected by the mean conditional probability of recurrence computed on the joint recurrence plots that are an extension of the close returns plots. This measure was successful in inferring the chaotic influence directions between nodes. By comparing the mean conditional probabilities, we were able to identify the source of the disturbance in the network.

With regard to the study of the spread of panic in emergency conditions, knowing the source of the propagation of a disturbance may be necessary to enhance the understanding of crowd movement and to anticipate the evacuation process. For future work, we aim to improve our approach to get further information on the interactions between the nodes more specifically on the direction of arcs. These information will

Once we'll be able to detect the sub-network formed by the chaotic nodes, we can think of another perspective which consists in studying the impact of the graph topology on the spread of a disturbance.

enable us to detect the chaotic network structure.

References

- Caldirola, D., Bellodi, L., Caumo, A., Migliarese, G., and Perna, G. (2004). Approximate entropy of respiratory patterns in panic disorder. *The American journal of psychiatry*, 161(1):79–87.
- Eckmann, J. P., Kamphorst, O. S., and Ruelle, D. (1987). Recurrence plots of dynamical systems. *Europhysics Letters*, 4:973+.
- Koiller, J. and Young, L.-S. (2010). Coupled map networks. *Non-linearity*, 23(5):1121.
- Mindlin, G. B. and Gilmore, R. (1992). Topological analysis and synthesis of chaotic time series. *Phys. D*, 58(1-4):229–242.
- Pecora, L. M., Carroll, T. L., Johnson, G. A., Mar, D. J., and Heagy, J. F. (1997). Fundamentals of synchronization in chaotic systems, concepts, and applications. *Chaos: An Interdisci*plinary Journal of Nonlinear Science, 7(4):520–543.
- Pereira, T., Strien, S. V., and Lamb, J. S. W. (2013). Dynamics of Coupled Maps in Heterogeneous Random Networks.
- Phatak, S. C. and Rao, S. S. (1995). Logistic map: A possible random-number generator. *Phys. Rev. E*, 51:3670–3678.

- Rabarimanantsoa, H., Achour, L., Letellier, C., Cuvelier, a., and Muir, J.-F. (2007). Recurrence plots and Shannon entropy for a dynamical analysis of asynchronisms in noninvasive mechanical ventilation. *Chaos (Woodbury, N.Y.)*, 17(1):013115.
- Romano, M. C., Thiel, M., Kurths, J., and Grebogi, C. (2007). Estimation of the direction of the coupling by conditional probabilities of recurrence. *Phys. Rev. E*, 76(3):036211.
- Rosenstein, M. T., Collins, J. J., and Luca, C. J. D. (1993). A practical method for calculating largest lyapunov exponents from small data sets. *PHYSICA D*, 65:117–134.
- Shibata, T. and Kaneko, K. (2003). Coupled map gas: structure formation and dynamics of interacting motile elements with internal dynamics. *Physica D: Nonlinear Phenomena*, 181(3-4):197–214.
- Trulla, L. L., Giuliani, A., Zbilut, J. P., and Webber, C. L. (1996).Recurrence quantification analysis of the logistic equation with transients. *Physics Letters A*, 223(4):255–260.
- Wolf, A., Swift, J. B., Swinney, H. L., and Vastano, J. A. (1985). Determining lyapunov exponents from a time series. *Physica*, pages 285–317.
- Zou, Y., Romano, M. C., Thiel, M., Marwan, N., and Kurths, J. (2011). Inferring Indirect Coupling By Means of Recurrences. *International Journal of Bifurcation and Chaos*, 21(04):1099–1111.

Pre-template Metabolic Replicators: Genotype-Phenotype Decoupling as a Route to Evolvability

William Hurndall¹, Richard Watson¹ and Markus Brede¹

¹University of Southampton, Southampton, UK W.Hurndall@soton.ac.uk

Abstract

The RNA World is widely heralded as the leading candidate for a template first vision of the origin of life, yet doubts as to the plausibility of the natural formation of RNA with catalytic function have led to revived interest in the metabolism first paradigm. Recent studies of the evolvability of reflexively autocatalytic sets of polymers have also revealed the nature of limited heredity in compartmentalised reaction networks. In the algorithmic sense, the lack of a meaningful distinction between data storage and functional expression results in a protocell heredity-fitness dichotomy which gives an intrinsic selective advantage to those protocells with limited heredity. An idealised model is used explore a minimal set of dynamical requirements necessary to weaken the dichotomy. This is achieved by explicitly modelling a protocell as an outer compartment 'phenotype', heritable only indirectly, in which competitive exponential growth may occur without compromising the heritability of previously discounted non-competitive growth autocatalysts in an inner sub-compartment, 'genotype'. Results show that heritable variation can be achieved under simulations of natural selection in populations of such metabolic replicators.

Introduction

Evolution by natural selection as proposed by Charles Darwin represents the founding piece and current cornerstone of the modern synthesis - the most complete description of how populations of organsims accumulate adaptation. However, because the mechanisms and units of heredity, variation and selection are considered to be static, the rigidity of the framework that gives it its explanatory power also prevent it from addressing the problem of just how nature came to adopt the Darwinian machine (Calvin (1997)) in its current form. A more complete model of the evolutionary process must then include a description of how the above mechanisms evolved (Pigliucci (2008)). Such evolution of evolvability is essential for the major transitions in evolution (Smith and Szathmary (1997)) and seems an especially relevant issue to consider in any candidate model for the origin of life. This problem is in part that of describing the emergence of a reproduction, variation and heredity dynamic (Maynard Smith (1986)). This criterion though

does not truly embody the algorithmic nature of the minimal machine believed to be requisite to the higher order evolution of evolvability and open-ended evolution. A key requisite considered here is the physical decoupling of the data storage device and the processing machinery responsible for functional expression and self-reference (Walker and Davies (2013); Ruiz-Mirazo et al. (2008)).

This paper is intended to summarise an ongoing investigation into the dynamical requisites of a purely metabolic replicator that could simultaneously permit non-trivial heredity, micro-mutation and the resulting phenotypes to be selectable/heritable. That is respectively, a parent replicator with non-trivial heredity must be able to transmit its phenotype to its progeny reliably, these phenotypes must be able to undergo small variation while they must also be selectable in a population. Together, this amounts to heritable variation. Using a minimal set of assumptions necessary, it is demonstrated that a duality between directly inherited autocatalytic molecules (genotype) and indirectly inherited molecules (phenotype) permits such heritable variation. In dynamical terms this corresponds to non-competitive autocatalytic growth in a protocell replicator's inner compartment and faster competitive growth in a disposable outer compartment. Algorithmically, this can be interpreted as the decoupling of the pseudo-instructional data storage and functional expression roles, reminiscent of aspects of geneprotein duality.

The first part of this paper summarises the spectrum of models of the origin of life. Then, an abstracted model of a metabolic replicator is introduced, designed to embody the key dynamical aspects of, and problems with, current models of compartmentalised metabolic reaction networks (protocells). In the model we present here, protocell fissioning dynamics are modified to distinguish fissionable material from non-fissionable (transmittable/non-transmittable to daughter protocells). This is designed to weaken the dichotomy, both allowing a kind of combinatorial heredity permitting micro-mutation, while also increasing the intrinsic fitness of these protocells that makes such variation heritable.

Template and Metabolism based Heredity

The two schools in origin of life research are broadly represented by the template first and metabolism first paradigms. The divergence is caused by the observation that both templating and metabolic function play distinct but intricately coupled and essential roles in all contemporary life. Linked genomes play the role of data storage that ultimately codes for protein enzymes, which elegantly guide biological metabolism, including the replication of the genome. In the current biological context then, each seems requisite to the other, giving rise to the chicken and egg paradox. The RNA World is the dominant candidate for a template first vision, in which naturally forming RNA strands contain sequential nucleotide data and also act as catalyst for their own base pairing replication (Gilbert (1986)). If they are able to multiply with heritable variation, then they may be able to evolve meaningfully under natural selection. Significant doubts have been cast as to whether any such polymer with both capacities, RNA or otherwise, could have formed naturally in the pre-biotic world (Bernhardt et al. (2012)). This kind of system relying on sequence based heredity stands in contrast to models of compartmentalised metabolic reaction networks that rely on attractor based heredity (Szathmáry (2006)). This is the distinction in the units of heredity between sequential instructional data that is internally inert until copied, and information contained as concentration profiles whose replication and hence heredity is the result of autocatalysis. Such models typically assume compartmentalisation of a repertoire of well mixed chemically autocatalytic units. These units are either autocatalytic at the level of the particle or some reflexively autocatalytic food generated set (RAF sets). These units would be potentially heritable as the numbers for each distinct unit type increase through autocatalysis until an enclosing compartment fissions into two. Multiplication and heredity at the compartment level results from the fissioning into daughter compartments, transmitting concentration profiles. Variation is desired as a consequence of the stochastic gain/loss of units during the growth and fissioning processes.

Alexander Oparin was the first to advocate the idea that spatially localised coacervates comprising varied chemical mixtures could have metabolised from the environment and become subject to natural selection. The first attempt to investigate the potential for heritability in metabolic models was taken by Farmer & Kauffman (Farmer et al. (1986); Kauffman (1986)). They studied artificial chemistries of protein polymers equipped with a capacity to catalyse ligation and cleavage reactions. Their aim was to investigate the conditions under which RAF sets of such polymers would form. They were motivated by the idea that when compartmentalised, the possible exponential growth of autocatalysts could both describe their own self-replication as units of heredity, and also confer exponential growth in compartment mass and hence exponential increase in compartment num-

bers. Selection would conceptually operate at the compartment level and those that increase in mass fastest would fission and multiply at a higher rate (have a higher Darwinian fitness). Compartments containing different RAF sets could then compete under natural selection. Others took this idea further, investigating the potential for heritible variation in these compartmentalised reaction networks when undergoing growth and fissioning cycles (Bagley and Farmer (1990); Bagley et al. (1992)). The idea was that if nature provided some form of spatial structure, such as naturally forming lipid micelles, vesicles, micro-spheres or coacervates, they could enclose the reaction network, preventing it from freely mixing within the environment. Other models have since used the same principles of imposed spatial segregation and autocatalysis, such as the GARD model of a Lipid World (Segré et al. (2000)) and Fernando & Rowe's model of chemical avalanches in reaction networks enclosed in liposomes (Fernando and Rowe (2007)). It seems that if such entities were in fact able to multiply with heritable variation, a set of models exist that assume a much lower level of early world chemical complexity than that of RNA like templating molecules. However there are in fact serious problems with limited heredity.

More recently, Vasas et al. (2012) have investigated the evolvability of the Farmer & Kaufmann type metabolic polymer networks. The essential results of the models were that only a single autocatalytic unit could typically be stably inherited by any single compartment and that heritable variation was in principle possible because stochastic processes could allow loss and gain of a compartment's complete repertoire of each unit. As Vasas et al. put it, this constitutes only one heritable bit of information. This does not seem to allow a meaningful concept of heritable micro-mutation, even if each unit as a set, had more than one attractor. The underlying problem is that the exponential increase in protocell numbers necessary for natural selection to act strongly at that level is caused by the necessarily independent and exponential growth of the chemically distinct units contained within. In general, this results in intra-compartment competition between units, which because of their exponential growth form inevitably results in only the fastest growing surviving within a compartment lineage; hence one heritable bit of information.

Model Aims and Methodology

This is not an investigation into the structure of reaction networks. The aim of this model is to identify a minimal set of physical and chemical requisites to heritable variation in a metabolic model of compartmentalised autocatalytic sets. As such, we focus on symbolic representations which we believe captures the relevant properties and problems. The ultimate aim would be to realise the model results in vitro, but here we aim only to use a theoretical tool to highlight the problems and identify a solution within the space of our

model. The purpose of doing this is to provide conceptual tools that might help elucidate the crux of the problem.

We wish to represent two extreme RAF set growth dynamics possible in artificial chemistries. One is that of unconstrained exponential/competitive growth (if no resource constraint). This is designed to approximate the growth form associated with autocatalytic reactions that is explored in most metabolic attractor based models. The other is that of some form of strongly self-regulating/non-competitive growth. The motivation for only including these growth types is that when compartmentalised they respectively represent the growth form that confers rapid exponential increase in compartment numbers (high intrinsic protocell fitness f), and the growth form that might best allow nontrivial heredity - non-trivial because an ensemble of distinct self-regulating units do not necessarily increase in number competitively. As such, in principle, they are capable of collectively and stably transmitting concentration profiles through a lineage as well as allowing additional such units to be lost or gained through stochastic processes, ie. allow multi-bit heredity. Summarising this point - protocells containing units that only grow competitively gives highly fit protocells, but poor heredity and those with units that grow non-competitively should give protocells with very low fitness but much better heredity. This represents an extreme tension between fitness and hereditary potential.

Non-competitive growth has been largely neglected in the literature on the evolvability of compartmentalised reaction networks for good reason. The protocell fissioning dynamics of current models in which the dichotomy exists uses a protocell which phase separates internal and external material in the environment only. In such cases if it is the protocell that selection operates on, the fittest compartment is the one that grows and hence multiplies fastest, not the one that has good hereditary.

What we aim to do is identify the minimal modifications to the protocell growth and fissioning dynamics that would break the tension between heredity and intrinsic fitness. For this purpose, we assume a two tier protocell system comprising an outer compartment which also houses an inner one. The inner compartment has much the same role as in other models - when a protocell (size of inner + outer) reaches a determined size, its contents are fissioned into two daughter inner compartments. However, during protocell growth, material is also allowed to leak from the inner into the outer compartment. Material in the outer is not directly transmitted to daughter protocells upon fissioning, but assumed washed back into the environment. The idea is that if an inner compartment can host non-competive units, when they leak to the outer compartment, they might either exhibit a competitive growth form due to different spatial constraints, or trigger competitive growth in the richer chemical environment of the outer compartment. Details of this dynamic are described in the following section, but the purpose of this modification is to reduce the large difference in intrinsic fitness between protocells whose inner compartment contains only non-competitive units, and those that contain competitive units by allowing both growth types to co-exist in different parts of a protocell with different function. The inner compartment could act as reliable data storage, while the outer does the bulk of the metabolising, providing the protocell level competitive growth.

Model

In our model, protocell growth and fissioning dynamics are similar to other models in which compartmentalisation of a chemical reaction network is imposed; the dynamics of a contained reaction network is run until the total particle numbers in the inner and outer compartment reaches F particles, then the inner compartment contents are stochastically fissioned into two daughter protocell inner compartments and next generation begun. The key difference is in the protocell structure. We assume a two tier structure in which only an inner compartment is fissionable into daughter compartments while the outer is lost to the environment during fissioning. During the growth phase, material is allowed to leak from the inner compartment to the outer where the chemical dynamics continue, as shown in figure 1.

We do not model a large reaction network, but work with an idealisation of only six types of autocatalytic unit, sacrificing rigour for clarity of principle and operation. We define three of them to grow in number exponentially at different kinetic rates in the inner compartment. These we call type A units. The other three are assigned the same rate constants, but growth forms designed to represent concentration dependant self-regulation in the inner compartment. These we call type B units. When in the outer compartment, both A and B types grow exponentially. The reason that we use three of each is that later we will test protocells containing each of the two types for heritable variation, and three is the minimum number necessary. An example of such a candidate self-regulating reaction is one in which a reactant catalyses its own production from some food set, but also catalyses the production of another particle that either inhibits the first reaction, destroys the unit or otherwise removes it from the system in such a way that both its growth and associated products grow non-competitively in mass. This type of reaction is reasonably well understood in chemistry and within the possibilities of a polymer chemistry such as that of Farmer & Kaufmann. Multi-particle examples are the Brusselator or Oregonator (Nicolis and Prigogine (1977)). It is not necessary for our purposes to describe in detail the many ways a reaction network could realise self-regulation in a many particle system, only to recognise the existence of reactions with either attractors or stable limit cycle solutions for the autocatalytic unit (or for many particle RAF set if autocatalytic unit is not a single particle autocatalyst). We model a single autocatalytic unit species and describe its

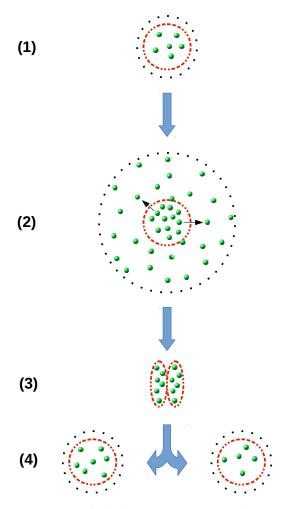


Figure 1: (1) Post-fissioning daughter protocell inner compartment. (2) Growth phase describes by chemical kinetics with uni-directional material leakage to the outer compartment with low volume density. (3) External agitation causes disruption to protocell when contained particle number reaches F. Outer material is washed away and inner compartment fissions each particle into one of two daughters with probability 0.5. (4) Two new daughter protocells are created and the process repeats from (1).

growth form as largely logistic. This was chosen to embody extreme non-competitive growth.

Chemical Kinetics and Protocell Growth Dynamics

The rate equations for i = 1,...N autocatalytic units with number x_i inside the inner compartment and y_i outside are used to model the growth of each type in a given protocell. In the following rate equations A type units are represented by species 1, 2, 3 and B type units by 4, 5, 6. Equations are numerically integrated and deterministic with integer valued particle numbers found by rounding the integrated equations at the end of a growth cycle.

Inner compartment kinetics:

$$\dot{x}_i = k_i \frac{x_i}{\left(\sum_{j=1}^N x_j\right)^p} - Dx_i$$
 $i = 1, 2, 3,$

(1)

$$\dot{x}_{i} = k_{i} \frac{x_{i}}{\left(\sum_{j=1}^{N} x_{j}\right)^{p}} \left(1 - \frac{x_{i}}{C_{i}(V_{I})}\right) - Dx_{i} \qquad i = 4, 5, 6.$$
(2)

Outer compartment kinetics:

$$\dot{y}_i = k_i y_i + Dx_i \qquad \qquad i = 1, 2, 3 \quad (3)$$

$$\dot{y}_i = k_i y_i + Dx_i$$
 $i = 1, 2, 3$ (3)
 $\dot{y}_i = k_i y_i \left(1 - \frac{y_i}{C_i(V_O)} \right) + Dx_i$ $i = 4, 5, 6,$ (4)

where $C_i(V) = c_i V/V_I$ and c_i is the carrying capacity for unit species i in volume V when p = D = 0. V_I is an assumed volume of the inner compartments and V is the volume of either the inner (V_I) or outer (V_O) compartment. Together with the assumption that $V_O/V_I \gg 1$, we treat the factor of the second term in equation [4], $y_i/C_i(V_O)$, as vanishingly small for the y_i ranges of the simulation. D is an approximation of a diffusion coefficient in the case of negligible outer compartment concentrations - we assume that concentrations in the outer compartment are low enough to ignore the actual internal-external concentration difference dependence of diffusion rates. It would also be desirable to represent some form of resource constraint on autocatalytic growth in the inner compartment that is most isolated from the wider environment and theoretical source of food species. Kaufmann, Farmer, Fernando & Rowe, Segre et al., all model resource constraint implicitly by assigning a rate at which food species leak into protocells. We have chosen to neglect detailed kinetics of such reaction dependencies, but would like to describe a monotonic reduction in the per particle resource as particle numbers increase. The ad hoc means by which we do this is to include a $1/(\sum_{j=1}^{N} x_j)^p$ factor for internal compartment unit growth. The effect of this is to apply an effective cost to growth in the inner compartment, modulated by p.

When placed in an inner compartment, the self-regulating B types will grow in a logistic like manner up to an effective carrying capacity (though not always $C_i(V)$ due to the resource constraint and D term). With the $V_O/V_I \gg 1$ assumption, they will diffuse into the outer compartment and begin true exponential growth. A type units will grow at up to exponential rates inside and truly exponentially outside. The rate equations basically do two things. First, they distinguish growth forms between the inner and outer compartments. As mentioned, this is to apply a cost to all inner compartment growth. The second, is to distinguish self-regulating growth of type Bs from type As, but only in the inner compartment. The $V_O/V_I\gg 1$ assumption gives them similar exponential/competitive growth in the outer compartment.

The reason that existing models do not properly consider autocatalysts that inhibit or remove themselves from the system, to be good candidates for units of heredity is because of the heredity-fitness tension mentioned in the previous section. That is, if a protocell were to contain only on phase seperated region, it would take far too long to reach a physical size at which it would fission, ie. it would be incredibly unfit. What allowing a non-fissionable outer compartment to do is be the medium in which self-regulators are allowed to change their growth form to competitive/exponential so that they can impart higher fitness upon their host compartment. As the growth form in the inner compartment is still self-regulating and the outer is not fissionable, this will not damage heritability of inner compartment concentration profiles so long as the protocell as a whole is still fit.

Assumptions

Many assumptions of the model are idealised representations of assumptions and results of existing models, such as the existence multiple autocatalytic sets in some chemistry, treatment of low particle number systems as well mixed, chemical dis-equalibrium caused by influx of energy rich food set and removal of material from the system, growth of compartment membrane material, and the compartment particle number dependence of the compartment fissioning parameter. The novel assumptions, also in idealised form are the following:

- There exist RAF sets that chemically self-regulate in a concentration dependant manner, which we approximate as single unit particulate autocatalysts.
- The spatial structuring in an environment is not described solely by a single phase separated protocell, but two phase separated regions, an inner and an outer, in such a way that a protocell's collective particle number contributes to compartment fissioning rates (intrinsic fitness f), but only the inner is fissionable.
- Outer and inner compartments have fixed volumes and $V_O/V_I\gg 1$.
- A $1/(\sum_{j=1}^{N} x_j)^p$ factor in the inner compartment kinetics describes a common resource constraint.

Recent work by Vasas et al. (2012) found that the number of exponentially growing autocatalytic cores (irreducible RAF sets (Hordijk et al. (2012))) was far exceeded by the number that were kinetically incapable of growing exponentially. It is in part from this result that we are motivated to assume the existence of self-regulators in this idealised

artificial chemistry. The assumptions with regard to a protocells's structural features are more ad hoc; notably the ability of the outer compartment to contribute to the physical instability of the inner compartment while being non-fissionable and at very low concentration. It is possible that multilamellar liposomes can provide some of this functionality, but this has not yet been explored.

We proceed with these as assumptions in this model while keeping the many alternative possibilities for future work, such as the ability of outer compartment autocatalysts to ignite other reaction pathways, which would not require the $V_O/V_I\gg 1$ assumption.

Hypotheses

- Combinations of self-regulating autocatalytic units can be faithfully transmitted down a lineage and protocells can undergo micro-mutation caused by stochastic loss or addition of distinct unit species.
- A two tier protocell in which outer compartment material is distinguished from fissionable inner compartment material will allow near unity intrinsic fitness f_B/f_A ratios of protocells characterised by self-regulating/non-competitive B type and non-self-regulating exponential A type units. This together with faithful transmission of concentration profiles between generations will permit heritable variation under simulations of natural selection.

Reducing the Intrinsic Fitness Disadvantage of Self-Regulators

If self-regulating units can be transmitted in combination, in order to demonstrate heritable variation, narrower intrinsic fitness differences between protocells containing only type Bs versus type As must be achieved. Otherwise the potential increased heredity allowed by self-regulators will not be heritable as artificial selective advantages assigned will not be great enough to overcome the intrinsic fitness differences. The first task was to identify which combinations of D, k_i, N_{max} and p would best reduce the intrinsic fitness differences of the protocells characterised by each type. The caveat was that after the growth phase, near complete regrowth of the B types in inner compartments would be necessary if combinations of distinct B types are to be heritable. If B types do not reach their effective carrying capacity in each growth cycle, then they will start to compete with one another within inner compartments within a lineage. This can be seen even in the logistic growth approximation for unit types 4,5,6; for particle numbers significantly below carrying capacity, growth is dominated by the first term in the kinetic equations, which potentially gives exponential growth (though will typically be of a parabolic/algebraic form due to the resource constraint assumption).

In all simulations in figure 2, the model was run through 100 growth and reproduction cycles to obtain a mean for

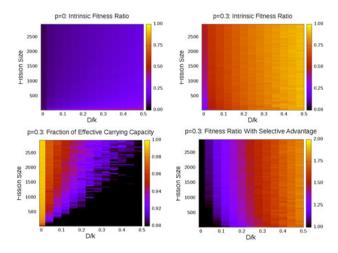


Figure 2: Selection of results from search of parameter space. The upper two plots show the ratio of the intrinsic fitnesses of B and A type compartments $f_B/f_A = T_A/T_B$ for p=0 and p=0.3. T_A and T_B are the mean times taken for protocellss containing A units 1,2 and those containing B units 4,5 to reach fission parameter F. Fitness is defined as their reciprocals. The lower right plot shows the same ratio when a selective advantage is given to the B types by multiplying the corresponding k_i rates by two. The lower left plot shows the total number of B types in an inner compartment prior to fissioning as a fraction of the maximum they can kinetically achieve - their effective carrying capacity.

the relevant observable. B and A type protocells were each seeded with two distinct B units (i=4,5) or two distinct A units (i=1,2). Two were used in each so that results would be relevant to later tests of heritable variation. The aim was to identify intrinsic fitness differences and measure the ability of B type protocells to achieve their effective carrying capacity IF a daughter protocell received its parent's full repertoire of unit types upon fissioning. For this reason, if a unit was removed entirely due to stochastic fissioning processes, it would be replaced with a single particle of that unit type.

The upper two plots in figure 2 show the disparity in the intrinsic fitness for p=0 and p=0.3. When p=0 the intrinsic fitness differences are so great for all D that complete regrowth is not particularly relevant because the necessary increase in fitness in any simulation of natural selection would be so great, they would be entirely unreasonable. When a cost to growth in the inner compartment is added by setting p=0.3, as D is increased the bulk of the growth starts to occur in the outer compartment where all unit types (i=1,2,..,6) grow exponentially and the intrinsic fitness differences are drastically reduced (at fission size the ratio of total particles in the inner to those in the outer is 0.041 for A type protocells and 0.006 for the B

type). The test for whether these fitness differences are great enough is whether increasing the fitness differences by multiplying the logistic type k_i values by a constant (in this case $k_i' = 2k_i$), will bring the ratio f_B/f_A to something greater than unity (changing the reaction rate constants k_i will be how we implement artificial selective advantages in simulations of natural selection for a target profile later). The lower right plot shows that this is the case. So long as this fitness is heritable, this compartment type will be able to compete effectively in a population. The lower left plot is an indication of how heritable such fitness differences might be for compartments of B types. It shows the contained number of particles in the given inner compartment prior to fissioning as a fraction of its maximum effective carrying capacity (separate simulation not shown). If this fraction falls below ≈ 0.9 , then the logistic growth starts to become exponential during growth phases, which means competition in the inner compartment for different values of k_i . The results show that low D around $D = 0.2k_0$, p = 0.3 and F = 2500 will provide the necessary theoretically possible fitness differences, while the effective carrying capacity fraction is high enough to indicate that the inheritance of these differences should be possible. See appendix for parameters.

Natural Selection and Micro-mutation -Heritable Variation

A crude test of heritability is to simulate natural selection acting on sets of protocells in a population. The following tests will allow us to evaluate at a high level whether previously imposed fitness ratios f_B/f_A approaching unity can be achieved in a given selection regime and whether the previously enforced transmission of an inner compartment's repertoire will follow from the earlier requirement that an effective carrying capacity is achieved during growth.

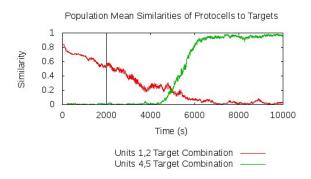


Figure 3: Mean similarity to 1,2 and 3,4 target in two simulations.

Two simulations were run. Stochastic addition and removal of a unit upon fissioning (other than the stochastic sampling process upon fissioning) was not allowed, but occasional migration of small randomly formed protocells con-

taining two particles was. In both, we seeded a population of $N_{compartments}=200$ protocells, each with a mixture of ten of the exponential units 1 and 2. After 2000 simulated seconds, selection on A type units 1 and 2 was initiated. The second simulation differs only in that selection was initiated for B type units 4 and 5.

Selection was implemented at each iteration of numerical integration by first assessing the similarity of a given protocell's total concentration profile with a target profile. The similarity measure used was the euclidean inner product between the normalised target and concentration vectors. Boltzmann scaling/selection was then used to evaluate a k_i' that was used to replace the normal k_i in the rate equations for all growth in that compartment for that integration step. Specifically, $k_i' = k_i(1 + \exp(\frac{\hat{\mathbf{T}} \cdot \hat{\mathbf{C}} - 1}{S}))$, where $\hat{\mathbf{T}}$ and $\hat{\mathbf{C}}$ are the normalised target vector and normalised concentration vector. Temporarily increasing the rates of reactions is a method used by others in simulations of natural selection for a target (Vasas et al. (2012)). Implementation here differs in that boltzmann scaling is used and the maximum artificial increase is 100% of k_i . See appendix for parameters and target vectors. When a protocell reaches F and fissioned into two daughters, one replaces its parent and the other replaces a randomly chosen protocell in the population.

Figure 3 shows the average similarity of protocells in the population to each target. Even though the populations were seeded favourably for the 1,2 target, the population similarity drops as migrant compartments containing species 3, the species with the highest k_i , invade the population and those already containing 1 and 2 fixate with only unit 2. Within protocell competition results in unit 3, with highest k_i , fixating within protocells and those protocells fixating in the population (as protocells containing only unit type 3 have the highest intrinsic fitness). Neither the targeted 1,2 or fitness is heritable. Invasion of the population by B type 4,5 protocells is possible however because their repertoire and fitness are heritable and so selection is able to induce greater exponential increase in their number until they come to dominance in the population.

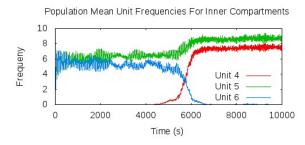


Figure 4: Population seeded with protocells containing unit types 1 and 2. No migration, but unbiased stochastic addition and removal of unit types 1,2 and 3 allowed.

These two simulations of natural selection confirm that there is heritability but do not confirm heritable variation, without which this model would not be demonstrating micro-mutation/heritable variation, but another means of canalisation. To test this, the previous simulation was rerun, being seeded only with unit types 5 and 6. Selection was continuous on unit types 4 and 5. Migration was disallowed as it was heritable micro-mutation/variation we were testing for rather than solely heritability under natural selection. Independent stochastic addition (P_{add} - probability of adding single particle at fission event) and removal (P_{remove} - probability of removing each particle in protocell at each fission event) of single particles of a random unit type of type 4,5 or 6 was introduced.

Figure 4 shows the integration step mean inner compartment particle frequencies. At approximately t=5000, the stochastic fissioning process admits a single particle of unit type 4 to a single protocell which also stochastically and independently loses all particles of unit type 6. Selection is then able to bring this compartment type/genotype to fixation.

Conclusions

The hypotheses that self-regulating/non-competitive autocatalytic units could permit faithful transmission of concentration profiles down a lineage while allowing micro-mutation and that a two tier protocell system would then permit heritable variation has been confirmed. This was achieved by identifying a two tier protocell system in which self-regulating autocatalysts were able to take the role of units of heredity, being directly transmitted to daughter protocells upon fissioning, while the bulk of the protocell's growth mass, and hence intrinsic protocell fitness, was achieved in an outer phenotype like region of the compartment where unconstrained exponential growth could contribute to fitness without harming heredity. The key physically meaningful parameters that influenced evolvability were diffusion rates D from the inner to the outer compartment, cost to inner compartment growth controlled by p and fissioning size F.

The solution to the weakening of the heredity-fitness dichotomy in this model has also coincided with an algorithmically meaningful distinction between data storage in the inner compartment and functional phenotype in the outer compartment. This was not an objective of the model, but a seemingly necessary consequence of minimal conditions for multi-bit heritable variation.

The limited heredity problem exists for both sequence/template and the attractor/metabolism based heredity mediums examined here. Ultimately, there is little experimental evidence to confirm that either RNA like molecules or RAF sets of protein polymers existed in the early world with the necessary high specificity catalytic/enzymatic function, and the problem of limited

heredity in attractor based mediums is even more severe. The model presented here represents one possible dynamical setting in which this problem might be alleviated. Additionally, the resultant 'phenotype', only indirectly heritable, represents the emergence of an algorithmically meaningful distinction between data storage medium and expression machinery. Given the increase in heredity, such a mechanism might also permit adaptations in the form of self-interactions, such as regulation of expression functions. While this possibility has not been explored within the scope of this simplified model, it is a unique feature with relevance to the evolution of evolvability. The evolution of development field has already demonstrated the use of self-regulation to achieve heritable adaptation in the units of phenotypic variation (Watson et al. (2014)).

Future work may include a more detailed treatment of the chemical and physical dynamics with examination of the feasibility of such a naturally occurring two-tier protocells and self-regulating autocatalytic units in the pre-biotic world. Of particular conceptual interest though is the potential of interaction between the inner and outer compartment networks, which might lead to the evolution of a protocell's ability to evolve. Additionally, while we believe we have presented the outline of a minimal dynamic potentially able to support open-ended evolution, we have not addressed the question of whether this system could itself have been the product of an evolutionary process. It is difficult to imagine the evolution of complex adaptations without good heredity, but the spatial structuring we assume here may not have required this to be an product of single-bit heredity (Powers et al. (2007)).

Appendix

Parameters and effective values used for simulations:

k_1	0.1	p	0.3
k_2	0.12	F	2500
k_3	0.14	D	0.02
k_4	0.1	V_I	1
k_5	0.12	V_O	∞
k_6	0.14	c_i	15 ∀ <i>i</i>
P_{add}	0.0001	$P_{migrate}$	0.005
P_{remove}	0.04	$N_{compartments}$	200
S	0.1		
$T_{1,2}$	$\frac{1}{2^{1/2}}(1,1,0,0,0,0)$		
$T_{4,5}$	$\frac{1}{2^{1/2}}(0,0,0,1,1,0)$		

Acknowledgements

This work was supported by an EPSRC Doctoral Training Centre grant (EP/G03690X/1).

Special thanks for helpful discussions with Anastasia Eleftheriou, Konstantinos Kouvaris and Simon Tudge.

References

- Bagley, R. J. and Farmer, J. D. (1990). Spontaneous emergence of a metabolism. Technical report, Los Alamos National Lab., NM (USA).
- Bagley, R. J., Farmer, J. D., and Fontana, W. (1992). Evolution of a metabolism. *Artificial life II*, 10:141–158.
- Bernhardt, H. S. et al. (2012). The rna world hypothesis: the worst theory of the early evolution of life (except for all the others). *Biol Direct*, 7(1):23.
- Calvin, W. H. (1997). The six essentials? minimal requirements for the darwinian bootstrapping of quality. *Journal of Memetics-Evolutionary Models of Information Transmission*, 1.
- Farmer, J. D., Kauffman, S. A., and Packard, N. H. (1986). Autocatalytic replication of polymers. *Physica D: Nonlinear Phe*nomena, 22(1):50–67.
- Fernando, C. and Rowe, J. (2007). Natural selection in chemical evolution. *Journal of theoretical biology*, 247(1):152–167.
- Gilbert, W. (1986). Origin of life: The rna world. *Nature*, 319(6055).
- Hordijk, W., Steel, M., and Kauffman, S. (2012). The structure of autocatalytic sets: Evolvability, enablement, and emergence. *Acta biotheoretica*, 60(4):379–392.
- Kauffman, S. A. (1986). Autocatalytic sets of proteins. *Journal of theoretical biology*, 119(1):1–24.
- Maynard Smith, J. (1986). *The problems of biology*, volume 144. Oxford: Oxford University Press.
- Nicolis, G. and Prigogine, I. (1977). Self-organization in nonequilibrium systems.
- Pigliucci, M. (2008). Is evolvability evolvable? *Nature Reviews Genetics*, 9(1):75–82.
- Powers, S. T., Penn, A. S., and Watson, R. A. (2007). Individual selection for cooperative group formation. In *Advances in Artificial Life*, pages 585–594. Springer.
- Ruiz-Mirazo, K., Umerez, J., and Moreno, A. (2008). Enabling conditions for open-ended evolution. *Biology & Philosophy*, 23(1):67–85.
- Segré, D., Ben-Eli, D., and Lancet, D. (2000). Compositional genomes: prebiotic information transfer in mutually catalytic noncovalent assemblies. *Proceedings of the National Academy of Sciences*, 97(8):4112–4117.
- Smith, J. M. and Szathmary, E. (1997). *The major transitions in evolution*. Oxford University Press.
- Szathmáry, E. (2006). The origin of replicators and reproducers. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 361(1474):1761–1776.
- Vasas, V., Fernando, C., Santos, M., Kauffman, S., and Szathmáry, E. (2012). Evolution before genes. *Biol Direct*, 7(1).
- Walker, S. I. and Davies, P. C. (2013). The algorithmic origins of life. *Journal of The Royal Society Interface*, 10(79).
- Watson, R. A., Wagner, G. P., Pavlicev, M., Weinreich, D. M., and Mills, R. (2014). The evolution of phenotypic correlations and developmental memory. *Evolution*.

A Robot that Uses Arousal to Detect Learning Challenges and Seek Help

Antoine Hiolle¹, Matthew Lewis¹, Lola Cañamero¹

¹ Embodied Emotion, Cognition, and (Inter-)Action Lab,
School of Computer Science & STRI, University of Hertfordshire,
United Kingdom
a.1.hiolle@herts.ac.uk

Abstract

In the context of our work on dyadic robot-human (caregiver) interaction from a developmental robotics perspective, in this paper we investigate how an autonomous robot that explores and learns novel environments can make use of its arousal system to detect situations that constitute learning challenges. and request help from a human at points where this help is most needed and can be most beneficial. In a set of experiments, our robot learns to classify and recognize the perceptual properties of various objects placed on a table. We show that the arousal system of the robot permits it to identify and react to incongruent and novel features in the environment. More specifically, our results show that the robot identifies perceived outliers and episodic perceptual anomalies. As in the case of young infants, arousal variations trigger regulatory behaviours that engage caregivers in helping behaviors. We conclude that this attachment-based architecture provides a generic process that permits a robot to request interventions from a human caregiver during relevant events.

Introduction

Since the pioneering work of John Bowlby regarding the role and the dynamics of attachment behaviors in development of infants from an early age (Bowlby, 1969), much research work in Developmental and Comparative Psychology has been devoted to address the impact of attachment dynamics on the socio-cognitive and emotional development of infants, both in human and non-human primates (see e.g., (Cassidy and Shaver, 2008; Bard et al., 2014)). As Bowlby highlighted following his observations of behaviors and affective displays in infants, a primary attachment figure - often the mother - plays a central role in regulating and orienting the infant during stressful periods, and during playand learning-oriented interactions. Bowlby developed a control systems theory of this aspect of social interaction, where proximity-seeking regulatory behaviors are produced by the infant as a response to the distress felt. These proximityseeking behaviors serve to attract and maintain the attention of the caregiver, and also help to regulate the developing emotions of the infant (Sroufe and Waters, 1977). The strategies employed by infants to attract and maintain the

attention of the caregiver and for emotion regulation are different in their nature and time line, and have been classified into different types of attachment profiles (Ainsworth and Bell, 1970). Bowlby introduced the notion of Secure Base to reflect the role that caregivers play in grounding the exploratory behaviors of infants, qualifying the attachment figure as an affective safe haven from which to explore (Bowlby, 1988) and develop social, affective and cognitive skills in a successful manner. Since then, much work has been devoted to the study of the factors responsible for the emergence of these patterns of behaviors, and the impact of the behavior of the attachment figure in their development (Mikulincer et al., 2003). Particular attention has been paid to the notions of Sensitivity - the ability to correctly interpret the behavior and demands of the infant – and of responsiveness - the timeliness of the responses of the caregiver which have been related to presence or absence of physical and emotional availability of a caregiver, and are thought to be determinants of the affect of the infant (Field, 1994).

The notion of arousal has been used in psychology to measure and quantify states of heightened activity, alertness, and attention, and was originally believed to reflect the activation of part of the central nervous system. This notion lead Hebb to propose a theory of drives based on an arousal system (Hebb, 1966). Moreover, arousal (together with valence) is considered one of the core dimensions of emotions (Russel, 1980). Alongside Hebb's work on the relationship between arousal, drives and goal-oriented behaviors, Berlyne postulated in his theory of curiosity (Berlyne, 1969) that low levels of arousal trigger exploratory behaviors whereas internal conflicts between expectations and the stimuli perceived give rise to a higher level of arousal. He added that the exploratory behaviors serve to promote a medium-to-optimal level of arousal. Berlyne hypothesized that arousal was a construct relating to "collative variables" and related them to exploratory behaviors as follows: "... [but] the paramount determinants of specific exploration are, however, a group of stimulus properties to which we commonly refer by such words as 'novelty', 'change', 'surprisingness', 'incongruity', 'complexity', 'ambiguity', and 'indistinctiveness'." (Berlyne, 1965, page 245). Furthermore, Berlyne formulated the notion of arousal as "all the stimulus properties that go to make up arousal potential, including the "collative" properties, e.g., novelty, variability, surprisingness, complexity, and ambiguity." (Berlyne, 1969, page 1068).

Our work in the area of developmental robotics has used the concept of collative variables to drive and regulate robot behaviours using variants of an architecture inspired from attachment theory. In previous studies, we showed how humans interact with a robot endowed with such architecture (Hiolle et al., 2012), and in more recent work (Hiolle et al., 2014), we demonstrated that the parameters of the architecture yield different patterns of exploration depending on the complexity of the environment. While these previous studies focused on global behavioural consequences of the use of collative variables regulated by human interactions, in this paper, we investigate in which specific situations these variables provide meaningful information and reflect the potential need for a human intervention. As these variables reflect the real time novelty or ambiguity of a given perceptual context, we need to be able to sample the realm of contexts and time span in which these variables are meaningful. More precisely, during an exploration and knowledge gathering episode, we investigate how and when does the dynamics of these variables offer an opportunity for a human to help development and learning. As our results suggest, human help can for example be useful to indicate to the robot whether to discard or attend to the current object of attention, or to provide physical assistance (e.g., moving the objects to help disambiguate perceptions) if the current situation is beyond the capabilities of the robot.

Related Work in Robotics

Arousal. Arousal has been used in artificial and robotic systems for different purposes. It has for example been used as a parameter to control the emotional displays of a robot as a function that reflects the levels of external stimulation received by an agent (Breazeal, 2003). Ogino and colleagues (Ogino et al., 2013) propose a motivational model of early parent-infant communication. Their model is based on the need for relatedness and its relationship to the dynamics of the pleasure and arousal in face-to-face interactions. They tested their architecture using a virtual robot on a computer which interacted with a human playing the role of the parent. To that end, their model includes a two-dimensional vector of pleasure and arousal following the circumplex model of emotions introduced by Russel (1980). The arousal of the agent is computed with respect to measures of novelty, stress and the perceived arousal of the human. The pleasure varies proportionally to the pleasure perceived, the relatedness, and the expectancy of the perception of some emotion in the human. Their study intended to reproduce the phenomenology observed during mother-infant interactions

and especially during still face episodes (Nadel et al., 2005). These episodes are characterized by a decrease in pleasure and positive emotions when the attachment figure stops responding to the infant's positive signals, such as gazing and smiling. The results they present show that this model reproduces the typical drop in positive affect following a still-face episode. Although the architecture based its novelty on a predictive system learning the likeliest next action the caregiver would produce, the interplay between the behavior of the caregiver and the exploratory behavior and learning of the robot were not studied.

The Attachment System. In the few studies trying to model the attachment system and its dynamics, the behaviors related to attachment and their occurrence are studied in isolation from other important facets of (infant) development. Typically, the socio-cognitive development is left aside, the attachment subsystem is considered on its own, and the analysis is solely concerned with the success or failure of a coping strategy or a regulatory behavior. For instance, Petters Petters (2006) presents simulations of caregiver-infant interactions using several control architectures based on attachment theory. The main goal of these simulations of artificial agents interactions was to model the relationships between the goals and behaviors observed in young infants. The resulting architectures were tested in unsafe or safe (secure or insecure) scenarios. Depending on parameters relating to the sensitivity of the caregiver of the infant agent, the behavior of the infant would vary. Specifically, the architectures comprised several main components inspired by the literature on Attachment theory. First, an Anxiety internal variable increases when the perceptual appraisal of the situation was deemed unfamiliar or unsafe. A Warmth internal variable was introduced to evaluate the positive interactions with the caregiver as hypothesised in the Secure Base paradigm. Based on these internal variables and the current perceptions, the action selection system assigns weights to the current goals and a winner-take-all approach is used to trigger the behavior associated with the most active one. Several variations of this architecture have been tested to include learning and adaptation from previous interactions. This adaptation was based on the success or failure to regulate the internal variables. For instance, the agent tries to approach its caregiver when the Anxiety variable is high, and the responsiveness or sensitivity of the carer (a built-in constant in the simulation) defines if the carer will provide Warmth and relieve the Anxiety. The reported results clearly show some emergent categories which are believed to correspond to the ones Ainsworth brought to light (Ainsworth et al., 1978). However, the attachment behavior itself is considered aside from the exploration and its potential consequences on development. In contrast with the models developed and tested in simulations in various studies concerning the emergence of attachment patterns (Petters,

2006; Stevens and Zhang, 2009), we have studied the dynamics of the dyadic interactions in a robot-centric manner. Our main aim is to improve the adaptivity of autonomous robots in order to, on the one hand, support their autonomous learning as a function of its interactions with the physical and social environment, and on the other hand improve the affective experience of the human in human-robot dyadic social interactions. However, despite the differences with the other body of work that models attachment dynamics and arousal modulation, we share the common view of the basic interplay between caring styles and behavior variations in affective adaptation.

Exploration, Curiosity and Intrinsic Motivation. A growing body of work in the robotics research community has focused on applying Berlyne's concept of curiosity as an intrinsic motivation for developing skills in robots. Following the encouraging results from the "playground experiment" from Oudeyer et al. (2007) and the advances in self-assessment measures related to novelty and learning progress (Şimşek and Barto, 2004), research has been devoted to the improvement of exploratory behavior and selfdevelopment of autonomous agents and robots. Most often these architectures use some evaluation of the progress of the agent in terms of learning, computed as the decrease of the prediction error of the Learning System of the robot (Kaplan and Oudever, 2005). Typical architectures modeling curiosity aimed at guiding the exploration of a developing robot often focus on specific task learning problem (Kaplan and Oudeyer, 2005; Luciw et al., 2011) and do not take advantage of the potential availability of humans. However, this principle has also been successfully applied to influence and help a robot in navigation tasks (Jauffret et al., 2013). In this contribution, the authors use self-evaluation measures of success and failure for the robot to express its "frustration" and trigger the help from a human when frustration is too high. They show how this strategy can help the robot subjectively identify deadlock situations, and be assisted in solving a given problem with the help of a human. In a similar fashion, a Surprise-based learning architecture was used to help a robot autonomously and incrementally build logical rules concerning the perceptions it meets in its environment (Ranasinghe and Shen, 2009).

Robot Architecture

In our previous experiments (Hiolle et al., 2012) an Aibo robot had to explore and learn the objects present in an environment consisting of toys and objects of different colors, shapes and sizes placed on a children's playmat inspired by the playground experiment (Oudeyer et al., 2007). The objects were a source of arousal for the robot as a function of their novelty. The level of arousal interacted with the parameters of the learning system, also influencing the observable behavior of the robot, particularly the time it would spend

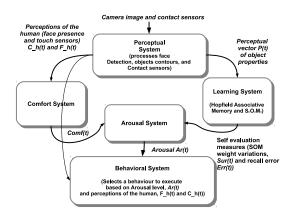


Figure 1: Components of the robot architecture used in the experiments. The Perceptual System processes the perception into a binary perceptual vector to be learned by the Learning System. We then evaluate the Stimulation as in Eq. 1, which is used with the *comfort* (Eq. 3) to compute the Arousal level in Eq. 2. The *arousal* level is used by the Behavioral System to choose the action to be performed.

in front of an object while learning it. In the experimental setup used in the study reported in this paper, the "task" of the robot was similar - exploring and learning novel objects in an environment, for which it can "solicit" (or not) the attention of a human carer as a function of its level of arousal caused by the exploration of the environment, and the carer can provide "comfort" via the visual (by showing his / her face) or tactile (patting the robot on the touch sensor placed on its head) modalities – but have we varied the previous setup as follows. We have used an Aldebaran Nao robot controlled using the architecture in Fig. 1 described in details below. The robot is placed in front of a table on which several colored objects (toy rubber cubes and balls) are placed as depicted in Fig. 2a. Following the previously presented body of work on attachment and exploratory behaviour for autonomous robots, we attempt to validate our architecture for attachment and dyadic exploration. We wish to evaluate when and how specific arousal variations occur and to which event they correspond to. In turn, this study will inform us on the specific helping behaviour a human can provide in this context, helping to generalize the results to different exploration and learning scenarios.

Perceptual System. The Perceptual System of the robot uses the image from the camera and the contact sensors located on the head of Nao to process information about the objects and humans around it. Perceptions feed into two different components of the architecture – the Comfort System and the Learning System. Perceptions about objects are extracted from the camera image and provide input to the Learning System. To perform visual perception of objects,

the Perceptual System extracts the contours in the image for the robot to learn features of the visual scene. To this end, we have used available visual processing tools from the OpenCV library. Our algorithm then selects the three largest closed contours using a Canny filter and extracts the following information from them. For each contour, the following properties are calculated to construct a binary vector P(t). The size of the area enclosed in the contour is measured as an integer in the interval [0, 5000], the length of the perimeter of the contour is evaluated as an integer in the interval [0, 1000], the average of the three colour channels in the RGB colour space is computed for the enclosed area of the contour, resulting in 3 floating point values in the range [0, 255]. The five values resulting from the previous steps are then normalised and discretized into 50 bins to construct a vector of 250 binary components P(t) which is used as input to the Learning System. Perceptions concerning human interventions might come from the camera or the contact sensors and provide input to the Comfort System and the Learning System. To be able to process the input from the human, the Perceptual System contains variables related to the presence of a face in the visual field $(F_h(t))$, and the values of the contact sensors $(C_h(t))$ located on the head of Nao. The presence of the face is a binary signal updated using the available face detection algorithm from the OpenCV library. The three contact sensors located on the head of the robot are also binary sensors, and are accessed and read using the URBI middleware.

Learning System and Self-Evaluation Measures. As in the architecture used in our previous work (Hiolle et al., 2012; Hiolle and Cañamero, 2008), the robot learns selected features of the scene using two neural networks – a Hopfield associative memory (Davey and Adams, 2004) and a selforganising map (Kohonen, 1997). These two types of learning algorithms were chosen for the following two reasons: first, their dynamics are well understood, and second, each algorithm provides two different but complementary capabilities associated with the task of learning: classification and recall. The self-organising map tries to classify the current binary vector, and the Hopfield network converges to the pattern closest to the input vector. At every time step, a new perception vector P(t) is presented as an input to the two networks, and an iteration of update and learning is performed by both neural networks. A real-time measure of the performance of the two networks is produced and stored as an internal variable named Stimulation, Stim(t) in equation 1, which is used to increase the level of arousal. Stimulation is computed as the half of the sum of the discrepancy between the pattern recalled by the Hopfield network Err(t)(recall error), and the sum of the variation of the weights of the self-organising map Sur(t).

$$Stim(t) = \frac{Err(t) + Sur(t)}{2} \tag{1}$$

$$\text{with } \left\{ \begin{array}{l} Err(t) = \sum\limits_{i=1}^{N} \mid S_i - P_i \mid \\ \\ Sur(t) = \sum\limits_{i=1}^{M} \sum\limits_{j=1}^{N} \mid w_{ij}(t) - w_{ij}(t-1) \mid \end{array} \right.$$

In equation 1, the intermediate variable Err(t) is the discrepancy between all N components of the recalled pattern from the Hopfield network (S_i) and the current pattern (P_i) , N is the number of components of the input vector, and M the number of units in the self-organising map. The value of Err(t) indicates how novel the current perception P(t) is for the memory, since the more novel it is, the higher the recall error Err(t) will be. The second term, Sur(t), is the sum of the variations of the synaptic weights of the Kohonen map. This "surprise" value reflects how far the current synaptic weights of the winning unit are from the current perception. This measure also reflects the novelty of the current perceptions P(t). These two measures are related to the prediction error used in other systems for self-evaluation (Weng, 2002).

The Arousal System and the Comfort System. The arousal model is an adaptation of the model described and studied in previous work (Hiolle et al., 2012). The arousal level increases as a function of the Stimulation perceived, to reflect the cognitive effort demanded by the current situation and the familiarity of the current perceptual vector P(t). The arousal is modeled as a smooth average of the Stimulation (see Eq. 1), which is a real-time evaluation of the recall error of the associative memory and the variation of the synaptic weights of the self-organising map. Additionally, in the same way as arousal and distress are modulated by the attachment figure in infants, the robot's caregiver can decrease the arousal via tactile contact or by presenting his/her face in the visual field, using the comfort computed from Eq. 3.

$$Ar(t) = \begin{cases} \frac{\tau_{ar} \cdot Ar(t-1) + Stim(t)}{\tau_{ar} + 1} & \text{if } Comf(t) \le 0.1\\ Ar(t-1) - \alpha_{ar} \cdot Comf(t) & \text{otherwise} \end{cases}$$
(2)

As we can see in Eq. 2, the arousal level is a scalar value computed as an exponential average of the stimulation perceived when no comfort Comf(t) is perceived. Exponential averaging is used to prevent sudden changes that could lead to abrupt changes in the behavior of the robot. The window parameter τ_{ar} controls the influence that the current Stimulation has on the arousal, thus defining its slope; it is a smoothing factor that biases this influence either towards "the past" (a larger τ_{ar} that produces smoother behavior) or towards "the present" (a smaller τ_{ar} that gives rise to more reactive behavior), as a function of the variability of the Stimulation.

Comf(t) is the internal variable evaluating the influence of the human, i.e., the comfort provided through the two available modalities, which are the perception from the head contact sensors $(C_h(t))$ and the perception from the face detection module $(F_h(t))$. Comf(t) is calculated as follows:

$$Comf(t) = \begin{cases} \frac{Comf(t-1) \cdot \tau_h + C_h(t) + F_h(t)}{\tau_h + 1} \\ \text{if } C_h(t) > 0 \text{ or } F_h(t) > 0 \\ \beta_h \cdot Comf(t-1) \text{ otherwise} \end{cases}$$
(3)

The trace rate β_h controls the rate at which the past comfort subsists in the system and provides the architecture with a means to vary the duration of the effect of human intervention. τ_h controls the weight given to past perceived comfort as it defines the time window on which the comfort value is updated. Both parameters were used in our previous HRI study (Hiolle et al., 2012) to define the two robot profiles that react differently to human interventions.

Interactions between the Arousal System, the Comfort System and the Learning System. The interaction between the arousal-comfort interplay and learning are as follows. Following the models previously discussed, a moderate level of arousal fosters learning, while extreme (high and low) levels of arousal hinder learning. An excessively high level of arousal reflects lack of "stability" in the underlying neural networks, leading the robot to stop in front of the stimulus currently perceived (the source of arousal) and "call for help" (look for a human). An excessively low level of arousal reflects "boredom" (lack of stimulation, lack of novel input to the networks) that leads the robot to divert attention away from the current stimulus and explore in search of new ones. The fact that the level of arousal descends from high to medium (when the decrease is not directly produced by human comfort) indicates that the robot is learning new, "interesting" things, and in fact a medium level of arousal following a high level is a sign that the robot has learned something new. The fact that the level of arousal descends from medium to low indicates that the robot is perceiving stimuli that are already familiar and have low "interest".

Action Selection and the Behavioral System. The Behavioral System, takes inspiration from behavior-based-robotics approaches, particularly (Brooks, 1986; Arkin, 1998; Avila-Garcia and Cañamero, 2004) and contains a set of predefined behaviors to be executed depending on the current perceptions and the arousal level. Each behavior possesses its own activation level, which reflects the relevance of that behavior for the current situation and is computed based on the Arousal level and behavior-related perceptual information. In a similar vein to (Avila-Garcia and Cañamero, 2004), our simple architecture implements in ef-

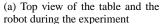
fect a two-resource action selection problem. Our robot must choose between two activities, "Explore-and-Learn" and "Find-a-Human" using a Winner-take-all action selection algorithm. The activation of these behaviors only depends on the level of arousal. If the arousal is greater than or equal to a given threshold (which here we have chosen to set to a high level, Highthresh), the behavior "Find-a-Human" will be executed. These two main behaviors can trigger other simpler behaviors, also following a Winnertake-all policy. The "Explore-and-Learn" behavior selects whether to attend to and learn the current stimuli ("Learn" behavior), or to move away from it and explore other elements of the environment ("Explore" behavior). The regulatory behavior "Find-Human" can either trigger the appetitive behavior to search for a face by moving its head (and therefore the camera located on its head), or the consummatory behavior of tracking a face (using the location of the face in the visual field provided by the perceptual system).

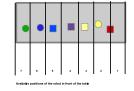
Experiments and Results

Experimental Setup

Arena. The arena used to carry our tests is shown in Fig. 2. Colourful objects are placed on a table covered with a black cloth to facilitate the extraction of the contours of the objects. The robot can then step laterally to change the view of the scene, and move the robot incrementally from one index position to the next. At the end of the table, the direction of the movement is changed, and it then starts moving in the other direction.



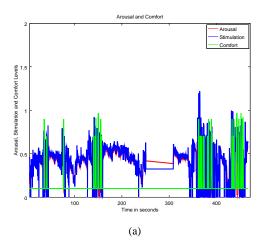




(b) Schematic of the top view of the table and the robot during the experiment with the possible positions and their labels

Figure 2: Experimental setup used with the Nao robot.

Caregiver Responses. In order to examine the architecture and the dynamics it produces in a systematic way, we designed an automated system to produce the responses of the caregiver. A "caregiving" response is produced every time the behavior "Find-Human" is activated and *arousal* is above the higher threshold, precisely one second after the behavior is activated, which is a good approximation (empirically established) to the time a human present by the setup takes to respond to the robot. The mechanism to produce



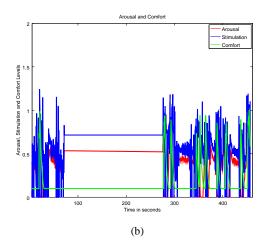


Figure 3: Graphs of the levels of Arousal, Comfort, and Stimulation during 2 typical runs (Arousal in red, Stimulation in blue, and Comfort in green). The robot is exploring the objects on the table, moving from position 1 to 7, and then back. After the robot has achieved this exploration twice, the experimenter swaps the objects for different ones (note that on both graphs, this corresponds to the time period where the values represented are constant). We can observe that during both these runs, once the objects were swapped, high arousal levels and more frequent comfort requests are recorded. This demonstrates how the system reacted to the modification of the environment.

this "caregiving" response consists of modifying the variables that monitor the presence of a human face $(F_h(t))$ and contact on the touch sensor on the head $(C_h(t))$, and hence to produce Comf. We summarise this process in Eq. 4.

$$\begin{cases} Req(t) = Req(t-1) + 0.1 & \text{if } Ar(t) > HightThresh \\ Req(t) = \alpha_c \cdot Req(t-1) & \text{otherwise with } \alpha_c = 0.1 \\ \text{if } Req(t) > 0.5 & F_h(t) = C_h(t) = 1 \end{cases}$$

$$\tag{4}$$

Tests and Results

The experimental runs unfolded as follows. First, the robot is placed standing in front of the table on the position labelled 1 in Fig. 2b. After the system is started the robot will attend current stimuli until the arousal felt drops below the low threshold. The robot will then move to the next position by stepping to the side. After the robot has walked in front of all the object twice, the experimenter pauses the system, and swaps the objects for new ones. The new objects vary from the initial ones either in shape or colour, for instance, blue cubes are replaced by purple cubes, and the small white cans are replaced by bigger ones. The robot's system is then resumed, and the robot continues its exploration and learning episode. During each run, we recorded all internal variables such as Arousal, Comfort, Stimulation, responses from the two neural networks (winner of the Kohonen Map, and recall error from the Hopfield associative memory). We also recorded all images used by the robot to extract the contours of the objects, and each contour was also recorded as an individual image to allow us to relate high Arousal episode to what the robot was perceiving and extracting at the time. Every run lasted between six and ten minutes. The parameters used in the experiment were the following: $\alpha_{ar} = 0.6$, the decay rate of Arousal, $\tau_{ar} = 5$, the time window for Arousal, $\tau_h = 3$,the time window for comfort, $\beta_h = 0.8$, the trace rate of comfort, HighThresh = 0.6, the higher threshold for Arousal, and LowThresh = 0.4, the lower threshold for Arousal. As we can observe in Fig. 3, when no comfort is provided, the Arousal level follows the trend of the perceived Stimulation, being computed as an average thereof. The first exploration of the table leads to high arousal episodes since the robot just discovers new objects after the other. Then, as can be seen in Fig. 3a after approximately 150 seconds, the arousal remains below the High threshold, when the robot has experienced the available features often enough. The same can be seen in Fig. 3b, after 60 seconds approximately. It has to be noted that the two presented runs are different in their timing, one robot took 200 seconds to go twice over the whole setup whilst one only took about 80 seconds. This can be due to a more favourable initialisation of the Kohonen Map synaptic weights and associative memory connectivity. Additionally, during the run presented in Fig. 3b, the robot faced less ambiguities and outliers (as can be seen in Fig. 4) than during the other run. After the objects were swapped, both runs show how differently the robot reacts (in terms of arousal felt and comfort requested) than in timesteps preceding the swap. The system reacts to these changes as expected when facing new objects and perceptual features, slowing down its exploration and learning further.

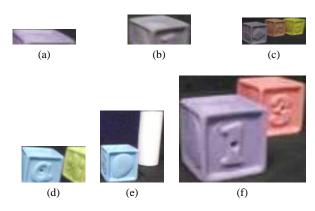


Figure 4: Samples of various outliers identified by the system. The following contours have been extracted following a high level of arousal during the experimental runs. We can see that these are examples of "failures" of the perceptual system in correctly extracting the contour of an object. Some of these examples include multiple objects extracted as on contour. These anomalies happen either when the objects are placed too close together or when the lighting conditions are less ideal. The algorithm might also extract the upper part of a cube if the light intensity is different between the upper and the frontal part of the object.

We show in Fig. 4 typical ambiguous perceptions that gave rise to high Arousal levels during the previous two runs. We can see that these cases mostly concern extracted contours that did not include a single object or highlighted some part of an object only. For instance, Fig. 4a and 4b show contours extracted that concern only the upper part of one of the rubber cubes on the table. These cases can happen due to changing lighting condition (when light is coming from above). Another common occurrence is several objects being extracted as one contour. This can be due to their too close positioning and having similar colours when represented in gray scale, which the algorithm for contour extraction uses. Moreover, this case would be occurring even more frequently if the robot were to try and manipulate the objects. When faced with these outliers, the robot triggers a regulatory behaviour to request help from a human. In a human-robot interaction, the human could choose to manipulate the objects to disambiguate the situation, for instance moving the object away from each other when they are too close together, or moving one object slightly closer of further so that the light is more evenly distributed on it. In this case, the human need not know which ambiguity stemmed the request, since he/she would not be able to know if the robot is perceiving only the upper part of the object, or if objects are too close. A natural -or common- behaviour observed in human is to move and shake objects in front of infants which in the case we present would be sufficient to solve the problem. Alternatively, the human could choose to make use of the comfort system, providing comfort via tactile interaction, and therefore lower the arousal and pushing the robot to explore further, discarding the current situation.

Conclusions and Perspectives

We have presented an architecture inspired from models of attachment that allows an autonomous robot to learn and classify perceptions of available objects and measure their relative novelty based on past experience. Based on these measures, the robot may trigger requests for help from a human in order to disambiguate or determine whether a situation is new and worth learning, or if some physical manipulation is needed from the human. The experimental results were gathered in a simple setting, where the robot was learning the features of colourful objects placed on a table in front of it. Furthering on results from previous studies, in this paper we have shown how and why a robot familiarizing itself with novel objects available in the environment would experience specific high arousal episodes, and how they related to the novelty of the environment and to specific ambiguities which arise in real-world interactions. We showed how the robot reacted to novel objects when the experimenter swapped the objects for new ones. In this instance, the robot reacted more strongly to these objects than previously, exhibiting more frequent regulatory behaviours. This type of regulation can be useful to notify the human that the environment or the current interaction has changed substantially, and giving the human the opportunity to pay closer attention and help the robot if needed, or intervene in order to solve a potential conflict that arose from the robot interacting with the environment. Alternatively, the human may also choose to redirect the robot's attention by providing comfort which promotes further exploration. The case highlighted in this study showed that the robot's perceptual system sometimes selected parts of objects or even multiple objects as contours to learn. These ambiguities could have arisen from how the robot manipulated the objects, variations in lighting conditions, or some other external influence, such as another human interfering with the setup. We therefore argue that this attachment-based system provides useful signals and ensuing behavioural dynamics for an autonomous robot engaged in exploration and learning. Exhibiting regulatory behaviours following these ambiguities could speed up the development and learning of the robot by making use of the human – her knowledge and help – in a timely fashion.

In the future, we plan to use of proprioception for the robot to explore its own capabilities and skills by touching and manipulating the objects, as a step towards engaging in physical interaction with the objects it learns.

Acknowledgements

This research was funded by the European Commission as part of the ALIZ-E project (FP7-ICT-248116). The opinions expressed are solely the authors'.

References

- Ainsworth, M. and Bell, S. (1970). Attachment, exploration, and separation: Illustrated by the behavior of one-year-olds in a strange situation. *Child development*, pages 49–67.
- Ainsworth, M., Blehar, M. C., Waters, E., and Wall, S. (1978).
 Patterns of attachment: A psychological study of the strange situation. Hillsdale, NJ: Lawrence Erlbaum.
- Arkin, R. (1998). Behavior-Based Robotics. The MIT Press.
- Avila-Garcia, O. and Cañamero, L. (2004). Using hormonal feed-back to modulate action selection in a competitive scenario. From animals to animats, 8:243–252.
- Bard, K., Bakeman, R., Boysen, S. T., and Leavens, D. A. (2014). Emotional engagements predict and enhance social cognition in young chimpanzees. *Developmental science*.
- Berlyne, D. (1969). Arousal, reward and learning. *Annals of the New York Academy of Sciences*, 159(3):1059–1070.
- Berlyne, D. E. (1965). *Structure and direction in thinking*. New York: John Wiley and Sons, Inc.
- Bowlby, J. (1969). *Attachment and loss*, volume 1: Attachment. New York: Basics Books.
- Bowlby, J. (1988). A secure base: Parent-Child Attachment and Healthy Human Development. Basic books.
- Breazeal, C. (2003). Emotion and sociable humanoid robots. *International Journal of Human-Computer Studies*, 59:119–155.
- Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23
- Cassidy, J. and Shaver, P., R. (2008). Handbook of attachment: theory, research, and clinical applications. Guilford Press.
- Davey, N. and Adams, R. (2004). High capacity associative memories and connection constraints. *Connection Science*, 16(1):47–65.
- Field, T. (1994). The effects of mother's physical and emotional unavailability on emotion regulation. *Monographs of the Society for Research in Child Development*, 59(2-3):208–227.
- Hebb, D. (1966). Drives and the cns (conceptual nervous system). *Brian Physiology and Psychology*, pages 67–83.
- Hiolle, A. and Cañamero, L. (2008). Why should you care? an arousal-based model of exploratory behavior for autonomous robots. In Bullock, S., Noble, J., Watson, R., and Bedau, M. A., editors, Artificial Life XI: Proc. of the 11th Int. Conf. on the Simulation and Synthesis of Living Systems, pages 242–248. MIT Press, Cambridge, MA.
- Hiolle, A., Cañamero, L., Davila-Ross, M., and Bard, K. A. (2012). Eliciting caregiving behavior in dyadic human-robot attachment-like interactions. ACM Transactions on Interactive Intelligent Systems (TiiS), 2(1):3.

- Hiolle, A., Matthew, L., and Cañamero, L. (2014). Arousal regulation and affective adaptation to human responsiveness by a robot that explores and learns a novel environment. Frontiers in Neurorobotics. Submitted.
- Jauffret, A., Cuperlier, N., Tarroux, P., and Gaussier, P. (2013). From self-assessment to frustration, a small step towards autonomy in robotic navigation. Frontiers in Neurorobotics, 7(16).
- Kaplan, F. and Oudeyer, P.-Y. (2005). The progress-drive hypothesis: an interpretation of early imitation. In Dautenhahn, K. and Nehaniv, C., editors, Models and Mechanisms of Imitation and Social Learning: Behavioural, Social and Communication Dimensions. Cambridge University Press.
- Kohonen, T. (1997). Self-Organizating Maps. Springer-Verlag.
- Luciw, M., Graziano, V., Ring, M., and Schmidhuber, J. (2011). Artificial curiosity with planning for autonomous perceptual and cognitive development. In *Development and Learning (ICDL)*, 2011 IEEE International Conference on, volume 2, pages 1–8. IEEE.
- Mikulincer, M., Shaver, P. R., and Pereg, D. (2003). Attachment theory and affect regulation: The dynamics, development, and cognitive consequences of attachment-related strategies. *Motivation and emotion*, 27(2):77–102.
- Nadel, J., Soussignan, R., Canet, P., Libert, G., and Gérardin, P. (2005). Two-month-old infants of depressed mothers show mild, delayed and persistent change in emotional state after non-contingent interaction. *Infant Behavior and Devel*opment, 28:418–425.
- Ogino, M., Nishikawa, A., and Asada, M. (2013). A motivation model for interaction between parent and child based on the need for relatedness. *Frontiers in psychology*, 4.
- Oudeyer, P., Kaplan, F., and Hafner, V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2):265–286.
- Petters, D. (2006). Designing agents to understand infants. PhD thesis, School of Computer Science, The University of Birmingham.
- Ranasinghe, N. and Shen, W.-M. (2009). Surprise-based developmental learning and experimental results on robots. In *IEEE* 8th Int. Conf. on Development and Learning, 2009. ICDL 2009, pages 1–6. IEEE.
- Russel, J. (1980). A circumplex model of affect. *Journal of Personality and Social Psychology*, 39:1161–1178.
- Şimşek, Ö. and Barto, A. (2004). Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 95. ACM.
- Sroufe, L. and Waters, E. (1977). Attachment as an organizational construct. *Child development*, 48:1184–1199.
- Stevens, G. and Zhang, J. (2009). A dynamic systems model of infant attachment. Autonomous Mental Development, IEEE Transactions on, 1(3):196–207.
- Weng, J. (2002). A theory for mentally developing robots. In Proceedings of the The 2nd International Conference on Development and Learning, pages 131–140. IEEE.

Estimating the Energy Cost of (Artificial) Evolution

Alan F.T. Winfield

Bristol Robotics lab, UWE Bristol, UK alan.winfield@uwe.ac.uk

Abstract

This short discussion paper sets out to explore the question: what is the energy cost of evolving complex artificial life? The paper takes an unconventional approach by first estimating the energy cost of natural evolution and, in particular, the species Homo Sapiens Sapiens. The paper argues that such an estimate has value because it forces us to think about the energy costs of co-evolution, and hence the energy costs of evolving complexity. Furthermore, an analysis of the real energy costs of evolving virtual creatures in a virtual environment, leads the paper to suggest an artificial life equivalent of Kleiber's law – relating neural and synaptic complexity (instead of mass) to computational energy cost (instead of real energy consumption). An underlying motivation for this paper is to counter the view that artificial evolution will facilitate the technological singularity, by arguing that the energy costs are likely to be prohibitively high. The paper concludes by arguing that the huge energy cost is not the only problem. In addition we will require a new approach to artificial evolution in which we construct complex scaffolds of co-evolving artificial creatures and ecosystems.

Introduction

Despite more than 20 years of good progress in evolutionary robotics the most complex robots evolved to date are – if we are honest with ourselves – not very complex.

Yet, within the wider discourse on predicted advances in super intelligent robotic and AI systems leading (perhaps) to a technological singularity (Eden et al., 2012), there is frequently an assumption that artificial evolution will do much of the heavy lifting in their development. For instance Chalmers (2010), in his philosophical analysis of the technological singularity, writes:

If we produce a AI by artificial evolution, it is likely that soon after we will be able to improve the evolutionary algorithm and extend the evolutionary process, leading to AI+.

I believe the assumption that artificial evolution will facilitate the technological singularity (i.e. the development of an Artificial General Intelligence followed by an intelligence explosion) to be mistaken, for several reasons. This paper focusses on one: the energy cost of evolving complexity. My contention is that this cost is likely to be colossal and well beyond the resources that may be realistically available in the near or medium term future. Of course I am not suggesting that the energy cost of artificially evolving human-equivalent AI (AGI) will be directly comparable to the energy cost of naturally evolving humans from scratch. I do, however, contend that an estimate of the latter is a useful argument in countering the optimism that increasingly appears to characterise opinion in AI researchers (Goertzel et al., 2010).

This paper proceeds as follows. First is an attempt to estimate the upper and lower bounds of the energy cost of human evolution. Then a consideration of the energy costs of artificial evolution, proposing an Artificial Life version of Kleiber's law and speculating on how the energy cost might scale with neural and synaptic complexity. The paper concludes with a short discussion around the evolution of complexity, noting that artificial evolution will need new approaches which combine artificial selection and niche construction.

The energy cost of evolving humans

An upper bound Let us estimate an upper bound by considering the energy delivered by the Sun, neglecting geothermal energy sources since these probably account for a very small proportion of the energy used by natural evolution.

Photosynthesis has been estimated to capture about 3000EJ per year in biomass¹ (Myamoto, 1997). It is generally accepted that plants extensively colonised the land during the Devonian period. If we take the mid-point of that period, $\sim 390MYa$, then the total energy captured by biomass since then amounts to $\sim 1.17 \times 10^{12}EJ$. Field et al. (1998) showed that the sea accounts for about the same primary (photosynthetic) production as the land despite very different physical distribution and producers (marine phyloplankton and land plants), so let us assume 1500EJ per year were captured by the sea's biomass during the long period ($\sim 3B$

 $^{^{1}\}mathrm{Total}$ human primary energy use in 2010 was estimated as 539EJ.

years) of evolution prior to colonisation of the land. This amounts to $\sim 4.5 \times 10^{12} EJ$. Thus we can estimate the total Solar energy capture by the Earth's biomass since cyanobacteria started photosynthesising as $\sim 5.7 \times 10^{12} EJ$.

This represents an estimate of the total amount of energy *available* to natural evolution, to date. Furthermore, this was the energy available to evolve *all* living things that have ever existed, including humans. Of course not all of that energy was used to power evolution – some of that energy is captured and stored in hydrocarbon deposits – hence this estimate is an upper bound.

A lower bound A different approach will yield a lower bound. By considering humans and working backwards via a series of Last Common Ancestors (LCAs) we can trace an evolutionary path to single celled organisms. Dawkins coins the term *concestors*, and identifies 39 in The Ancestor's Tale (Dawkins, 2004). To simplify the energy estimate we choose a subset of concestors, punctuating the stages of evolution from hominids to primates, then mammals, land vertebrates, sea vertebrates, multi-celled animals and finally single-celled animals. For each of these seven stages we need three values: the average daily energy consumption of each individual, the average age of reproduction, and the average number of individuals in our ancestor's population at any given instant. Estimating these values for the more recent stages of evolution is relatively straightforward – especially for daily energy consumption and age of reproduction. For the energy consumption of an organism with a given mass we can refer to Kleiber's law (Dawkins, 2004, p. 422). Population size is much more difficult and here we can do little more than guess: too small a value and there is insufficient genetic diversity, whereas too large doesn't make sense if our ancestor's size and mobility meant it could only access a limited breeding group.

A key assumption of this lower bound estimate is that the species (orders) that branch away from our concestors are not directly implicated in the evolution of humans, because of symbiosis, mutualism or food-chain dependency. In other words we discount the energy cost of the continuing evolution of reptiles (including birds) after concestor 16 - the last common ancestor of mammals and reptiles. Uncontroversially we discount the energy cost of lungfish, from concestor 18, and of the ambulacrarians (including starfish, sea urchins and sea cucumbers) from concestor 25. More controversially we discount the energy cost of the subsequent evolution of insects (which branch from concestor 26) after 570MYa, and plants (which branch from concestor 36) after 900MYa. Humans (and many other ancestral species) have a food-chain dependency on plants and insects, and flowering plants depend on the mutualism of plant and (insect) pollinator. It is for these reasons that the energy estimate here is a lower bound.

Given the estimated (in some cases guessed) values in Ta-

ble 1 and summing the energy costs per epoch we arrive at a lower-bound estimate for the energy cost of evolving humans, of $\sim 8000EJ$. We shall return to the difficult question of where – between the lower and upper bounds estimated here – the true energy cost of evolving humans might lie.

The energy cost of artificial evolution

The evolution of robots (or AI) with human-equivalent intelligence almost certainly will not require that we recapitulate natural evolution from scratch, either in materio or in silico. But, from an energy perspective, that doesn't let us off the hook. Evolutionary robotics has, to date, mostly evolved controllers - often based upon simple artificial neural networks – for relatively simple pre-designed robots. The most complex robot controllers evolved to date have perhaps 100 artificial neurons, somewhat less but of the same order as C. elegans (nematode roundworm), with 302 neurons and ~ 5000 synapses. Such a controller is typically evolved in a simulated environment and then downloaded into the real robot. The process of artificial evolution may require a population of 100 individuals (genomes), each of which needs to be instantiated as a simulated robot and fitness tested in its simulated environment, for perhaps 1000 generations. If that environment models physics, as well as the robot's sensors and actuators with sufficient fidelity, then a workstation grade PC may complete the task in 10 hours, at a total energy cost of $\sim 9000 KJ$. Setting aside the fact that the robot's artificial neurons are generally very much simpler than C. elegans biological neurons, we have an energy cost estimate of evolving an artificial controller for a robot, of roughly comparable neural complexity.

Recent work by Auerbach and Bongard (2014) explores the influence of the environment on the evolution of morphological complexity in virtual machines, in work that is representative of the state-of-the-art in the co-evolution of morphology and control system. This elegant research demonstrates that increasing morphological complexity is actively driven by environmental complexity. This work provides further evidence of the energy cost of artificial evolution; the exploration of artificial organisms of low complexity (compared with biological organisms) evolved for one behaviour only – efficient locomotion across a ridged 'icy' landscape, is reported to have cost 100 CPU-years of computational effort on a supercomputing cluster.

One objection to the approach outlined in this paper is that there is no proper basis for comparison, no equivalence, between biological evolution and artificial evolution. Of course the processes and mechanisms are profoundly different (except for the meta-level equivalence of the Darwinian evolutionary operators: variation, selection and heredity), but there is an ineluctable truth: artificial evolution still has an energy cost. Virtual creatures, evolved in a virtual world, have a real energy cost. And we can estimate that energy cost. For the *C. elegans* equivalent example outlined above

Epoch	Hominid	Primate	Mammal	Land	Sea	Multi-celled	Single-celled
				Vertebrate	Vertebrate	organism	organism
Concestor no (Dawkins (2004))	1	9	16	18	25	32	_
Indiv. energy cost per day (KJ)	8500	4000	1000	100	100	10	0.0001
Time to reproduction (years)	15	5	0.2	1	1	1	0.003
Life to repro. energy cost (KJ)	46537500	7300000	73000	36500	36500	3650	0.0001
Epoch (M years)	6	20	150	200	200	100	3000
Generations per epoch	400000	4000000	750000000	200000000	200000000	100000000	1.095E+12
Population size per generation	1000	1000	10000	10000	10000	100000	100000000
Energy cost per epoch (EJ)	186.15	292	5475	730	730	365	109

Table 1: Lower bound energy calculation

each simulated robot has a real energy cost of about 9J/hr, which interestingly is about 2000 times greater than the energy cost of a very small (1mg) organism, 0.004J/hr. It is clear that 'larger' artificial creatures, i.e. with more artificial neurons, must incur a greater computational energy cost.

In general, if the energy cost of simulating and fitness testing a virtual creature is e, then the energy cost of evolving that creature will be E=gpe, where g is the number of generations required and p the population size. Energy cost e is clearly a function of the complexity of that virtual creature, but how might e scale with complexity? Kleiber's law (Dawkins, 2004, p. 422) relates the mass of an organism to its energy consumption and, plotted on logarithmic axes, shows a remarkably consistent linear relationship from micro-organisms to the largest animals. Perhaps a similar relationship might exist between, say, neural complexity and energy cost e for virtual creatures: an artificial life equivalent of Kleiber's law?

Figure 1 imagines such a plot, of neural complexity against energy cost e. We cannot yet plot such a relationship since we have, to date, only one or two points at the very bottom of the artificial neural complexity scale. But, if we assume that a human-equivalent AI will require roughly comparable neural complexity to *Homo Sapiens*², with 85×10^9 neurons and $10^{14} - 10^{15}$ synapses (and noting that neural complexity must take account of the number of synapses, since neural connections incur a computational energy cost), then e for an artificial creature of this synaptic complexity could be $10^{10} - 10^{12}$ times greater than for something equivalent to C. elegans. But this scale factor is still likely to be too low because fitness testing of increasingly complex artificial creatures will take longer and incur greater energy cost. It seems likely that the gradient of our ALife version of Kleiber's law will be greater than 1.

Discussion

An important consideration is the question of what, exactly, do we mean by the evolution of complexity. Levins and Lewontin (1985) articulate the significant difficulty of

measuring complexity and demonstrating its increase during evolution. Adami et al. (2000) explore the same question by developing an information theoretic approach to biological complexity. We should rightly be wary of any suggestion of a monotonic increase in complexity during evolution. And, returning to the energy cost of evolving humans, much of the structural and morphological complexity of hominids – of vascular and nervous systems, skeletons and sense organs – was established early in our evolutionary history. But then somehow the happy coincidence of dextrous hands with opposable thumbs, acute forward-facing binocular vision and big brains with a neocortex gave rise to a late explosion of phenotypic complexity in the last 6M years, resulting in what Mithen (1996) calls the architecture of the modern mind.

Attempting to estimate the energy cost of evolving humans, and establishing approximate upper and lower bounds on this cost, exposes a deeply interesting question: how much of the Earth's biota was necessary for the evolution of humans? Biological complexity apparently arises from an evolutionary arms-race in which organisms both adapt to and exploit niches in their ecosystem and – in so doing – co-create that ecosystem. As Levins and Lewontin (1985) point out the organism is both the subject and the object of evolution. Niche construction is the process by which organisms continuously modify their own and others' niches; from a niche construction perspective "evolution consists of mutual and simultaneous processes of natural selection and niche construction" (Laland et al., 2000). So, the answer to our question 'how much of the Earth's biota was necessary for the evolution of humans?' is something we cannot know, since unpicking the immense tangle of co-evolving species and niches is almost certainly impossible.

But what seems clear is this. To evolve artificial life, or AI, of significantly greater complexity than anything so far achieved will require a new approach to artificial evolution in which we construct complex 'scaffolds' of co-evolving artificial organisms and ecosystems. Learning how to sustain artificial scaffolds for long enough to make real progress will be a significant long-term challenge, requiring both ingenuity and energy. From an energetic point of view the Kleiber's law like relationship between neural and synaptic

²almost certainly not a safe assumption, but it's all we have to go on here.

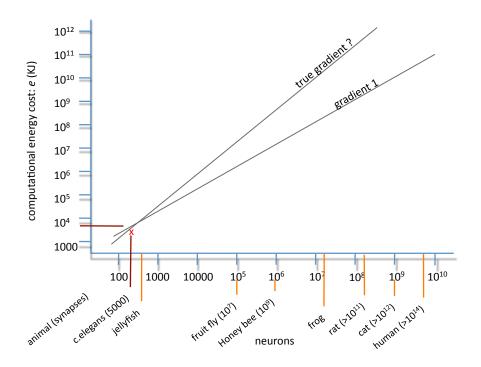


Figure 1: The computational energy cost of artificial neural complexity, after Kleiber's Law.

complexity and the energy cost of evolution suggested in the previous section is almost certainly a gross oversimplification. To evolve artificial creatures (or robots) of significant neural complexity will require that we co-evolve multiple 'species' within complete eco-systems in a parallel process of artificial selection and niche construction, in order to promote the evolution of greater levels of complexity and capability. Even if we succeed in understanding how to engineer such scaffolds, the energy costs are likely to be many orders of magnitude greater than Fig. 1 might suggest. Artificial evolution is not the silver bullet that advocates of the technological singularity might suppose it is.

Acknowledgements

The development of these ideas have greatly benefited from discussion with several people including Susan Blackmore, Jolyon Troscianko, Andy Radford and Marina Strinkovsky.

References

Adami, C., Ofria, C., and Collier, T. (2000). Evolution of biological complexity. *Proc. Nat. Acad. Sci. USA*, 97:4463–4468.

Auerbach, J. and Bongard, J. (2014). Environmental influence on the evolution of morphological complexity in machines. *PLoS Computational Biology*, 19 (1): e1003399.

Chalmers, D. (2010). The singularity: A philosophical analysis. *Journal of Consciousness Studies*, 17 (9-10):7–65. Dawkins, R. (2004). *The Ancestor's Tale*. Weidenfeld and Nicolson, London.

Eden, A., Steinhart, E., Pearce, D., and Moor, J. (2012). Singularity hypotheses: An overview. In Eden, A., Moor, J., Soraker, J., and Steinhart, E., editors, *Singularity Hypotheses: A scientific and philosophical assessment*, pages 1–12. Springer-Verlag, Heidelberg.

Field, C., Behrenfeld, M., Randerson, J., and Falkowski, P. (1998). Primary production of the biosphere: integrating terrestrial and oceanic components. *Science*, 281:1802–1804.

Goertzel, B., Baum, S., and Goertzel, T. (2010). How long till human-level AI? *H+ Magazine*, pages 237–240.

Laland, K., Odlimng-Smee, J., and Fledman, W. (2000). Niche construction, biological evolution, and cultural change. *Be-havioral and Brain Sciences*, 23:131–175.

Levins, R. and Lewontin, R. (1985). *The Dialectical Biologist*. Havard University Press.

Mithen, S. (1996). *The Prehistory of the Mind*. Thames and Hudson, London.

Myamoto, K. (1997). Chapter 2: Energy conversion by photosynthetic organisms. In *Renewable biological systems for alternative sustainable energy production (FAO Agricultural Services Bulletin - 128)*. Food and Agriculture Organisation of the United Nations.

Designing a minimalist socially aware robotic agent for the home

Matthew R. Francisco¹, Ian Wood¹, Selma Šabanović^{1,2}, and Luis M. Rocha^{1,2,3}

¹School of Informatics and Computing, Indiana University, Bloomington, IN 47406, USA ²Cognitive Science Program, Indiana University, Bloomington, IN 47406, USA ³Instituto Gulbenkian de Ciencia, Oeiras, Portugal francm@indiana.edu

Abstract

We present a minimalist social robot that relies on long timeseries of low resolution data such as mechanical vibration, temperature, lighting, sounds and collisions. Our goal is to develop an experimental system for growing socially situated robotic agents whose behavioral repertoire is subsumed by the social order of the space. To get there we are designing robots that use their simple sensors and motion feedback routines to recognize different classes of human activity and then associate to each class a range of appropriate behaviors. We use the Katie Family of robots, built on the iRobot Create platform, an Arduino Uno, and a Raspberry Pi. We describe its sensor abilities and exploratory tests that allow us to develop hypotheses about what objects (sensor data) correspond to something known and observable by a human subject. We use machine learning methods to classify three social scenarios from over a hundred experiments, demonstrating that it is possible to detect social situations with high accuracy, using the low-resolution sensors from our minimalist robot.

Introduction

In 2003, Rodney Brooks suggested that "by 2020 robots will be pervasive in our lives" (Brooks, 2002, p. 113). As an example of this trend, he conceptualized an autonomous robotic vacuum cleaner for the home with a bottom-up design which allowed the robot's behaviors to emerge in interaction with its physical environment. The robot used the amount of light it sensed as a measure of the dirtiness of the floor, readings from its bump sensors as a signal to change direction, and its cliff sensors to know when to stop so as not to fall down stairs. Without full knowledge of the physical environment, the robot could randomly cover yet fully clean a wide variety of floors. The *iRobot Roomba* robotic vacuum, commercialized in 2002, is the materialization of Brooks' idea; more than 10 million Roombas have so far been sold worldwide.¹

As a robust, commercially available robotic product, the Roomba was one of the first robots to be used naturalistic and long-term studies of human-robot interaction in the home. These pioneering studies ascertained that, along with the physical environment, the *social context* also had an effect on the cleaning robot's ability to function successfully. Researchers described that domestic Roombas were given names (Sung et al., 2007) and treated as "social agents" (Forlizzi, 2007); the use of Roombas also had a reciprocal effect on the social organization and practices in the home, as men and teenagers participated more in domestic cleaning chores. Such findings call attention to the importance of understanding the social as well as the physical dynamics of the context of use for robotic products.

Inspired by the Roomba as a commercial and social product, we propose that future robotic technologies that can coexist and collaborate with people in everyday environments should have a sense of the social as well as physical contexts in which they operate. Contemporary robots are largely ignorant of the social significance of their actions and of the bustle of human life around them. As robots spend more time around humans, they will profit from being able to take advantage of the social as well as the physical characteristics of the environment to support their successful functioning (Dautenhahn et al., 2002).

Artificial Life has contributed greatly to the development of situated robots whose behavior emerges from the nonlinear interaction between machine and environment (Almeida e Costa and Rocha, 2005). But just like human cognition and social intelligence is extended into the environment (Clark, 1998), robots can use the bottom-up principles of artificial life to develop social competency. Uexkull's concept of *umwelt* (the self-centered sensorial world of animals) has served as a guiding principle to generate robot behavior that is grounded on their own perception-action interaction with an environment (Hoffmeyer, 1997). While the concept of an *umwelt* makes the case for personal sensory experience, Uexkull (2001) notes that it gives us a way to understand sociality as an *intersubjective* process rather than a subject-object dualism:

...the idea of an objective universe, that embraces all living things, is undeniably very useful for ordinary life. The conventional universe, where all our relationships to our fellow human beings are enacted, has

¹http://www.irobot.com/en/us/Company/About.aspx?pageid=79

brought all personal Umwelt spaces under a common denominator, and this has become indispensable for civilized human beings. Without it, we cannot draw the simplest map, because it is impossible to combine all subjective points of view in a single picture (p. 109).

Similarly, we now need to develop social umwelten for robots. By this we mean that, rather than designing robotic social behavior in a top-down manner, we need to develop it from the bottom-up in a manner that is most consistent with the robots own sensors and its interaction with the social environment (including designers). This also means that the robot and humans must have more ways of recognizing, remembering, and building upon intersubjective experiences. Within this multi-agency environment the idea is to develop umwelt overlap between robotic and human agents that scaffolds cooperative action (Ferreira and Caldas, 2013).

We describe an initial approach to developing social awareness and presence for a domestic robot using minimalist robots, a combination of simple sensors, and bio-inspired computational techniques to develop a social umwelt for domestic robots. We seek to develop socially situated robotic agents whose behavioral repertoire is subsumed by the social order of the space. The goal is to recognize different classes of human activity and then associate to each class a range of appropriate behaviors. While human-robot interaction research has largely focused on developing algorithms that use high resolution data such as audio and video, our system relies on long time-series of low resolution data such as mechanical vibration, temperature, lighting, sounds and collisions. We rely on low-resolution data because that is the reality of the sensors in the robot platform we use (see below). This allows us precisely to test if such cheap sensors, which are immediately and widely available, are capable of developing minimal social awareness.

We begin the paper with a discussion of the Roomba and its relationship to the social and cultural models of the home. We then introduce the robot and test how it experiences the world through its sensors. These exploratory tests allow us to develop hypotheses about what objects (sensor data) correspond to something known and observable by a human subject. In the third section we use machine learning methods to classify three scenarios from over a hundred experiments involving human interaction. We conclude with some future directions for our research.

Navigating social spaces

The Roomba is one of the first instantiations of robots that work in everyday human environments with untrained users. Technology corporations and governments around the world expect that such technologies will proliferate and provide a new era of technological and economic production. This future direction for robot development is highlighted by the US National Science Foundation's National Robotics Initiative, which funds the development of co-robots that "work

beside, or cooperatively with, people... acting in direct support of and in a symbiotic relationship with human partners" (National Science Foundation, 2013). We use the Roomba as a model case for studying how robotic technologies might be socially integrated into human environments.

Roomba's design, inspired by Brooks' subsumption approach to artificial intelligence, provides a robust and workable solution to issues posed by diverse and constantly changing human environments. Studies of Roomba's use in actual homes, however, have pointed out that there are important challenges and resources in the environment that the Roomba's design does not take into account. For example, while the Roomba is designed to function in rooms of different shape, size, and organization, it is limited in the kinds of terrain it can cover. Therefore owners need to adapt the home to their Roombas by moving furniture, objects, and moving them between different levels of a house (Forlizzi, 2007). More relevant to our goals, Roomba's current limitations in terms of social awareness also limit its use; users may find the Roomba's random coverage of space may disrupt their activities, and therefore turn the Roomba on only when they are away from home or at specific times.

Existing research on robots in the home has shown that even functional robots like the Roomba are interpreted in social ways when they are situated in social environments (Forlizzi, 2007). The robot's emergent interactions with the social organization, cultural norms, political dynamics, and people's interpretations within the social environment can have a significant effect on whether the robot is accepted or rejected by users (Mutlu and Forlizzi, 2008). We have also shown that users envision and evaluate robots and other technologies in the context of the social hierarchies and relationships they regularly inhabit (Lee and Sabanovic, 2013). Initial research on service robots further suggests that personalization is an important component of robotic functions in open-ended human environments (Forlizzi and DiSalvo, 2006)

The idea that human interpretations of a robot's functioning can support its behavioral repertoire was suggested decades ago in Braitenberg (1984) description of robotic "vehicles" whose simple behaviors in relation to the environment evoke ascriptions of affective and cognitive meaning by human observers. Several social robotic projects, such as Keepon (Kozima et al., 2009), Muu (Matsumoto et al., 2005), and PARO (Shibata, 2012), have embraced the possibility of communicating social presence and agency through simple relational cues. Alač et al. (2011) has shown that the social agency of robots is not constructed solely, or even primarily, through their functional capabilities, but is scaffolded through the ways in which human actors orient themselves towards the robot.

This research in human-robot interaction corroborates prior theories in human-computer interaction that describe everyday contexts as not only physical, but social and cul-



Figure 1: Katie in operation.

tural spaces that hold both personal and shared meanings for their human inhabitants, which are constructed through embodied interaction (Dourish, 2001). We extend this understanding of how space relates to the design of robots and human-robot interaction by developing minimalist robots that are becoming aware of the social and cultural characteristics of their environment.

The Katie family social robot

In this section we give a description of our robots and the kind of sensorial world in which they operate. The graphs of the sensor data in this section are meant to demonstrate the flow of data from the robots that can be interpreted by most people. Can the robot tell the difference if it is next to a wall versus in the middle of a room? If a person greets the robot, which sensors exhibit change and how much? The graphs of this data are a resource of the robot's social umwelt as it is the first place where both humans (researchers) and our robots establish a common denominator together with an object in the world. Most importantly, graphs such as these help us to understand the limits of real time sensor responses, graphic visualization of data, and humans linking those to meaningful environmental changes. To translate further between robot and human will require more sophisticated techniques, which we discuss in the next section.

Robot design

The robots we are using, called the *Katie family*, are built on the *iRobot Create* platform, an *Arduino Uno*, and a *Raspberry Pi*. The Create has several native sensors and we have included more sensors to the Arduino (see Figure 2). The Raspberry Pi collects images which are used for coding data and verifying our classes. It also relays data to and commands from our server. Our research group can begin and start experiments remotely and download data from each experiment. These basic parts comprise the embodiment of the

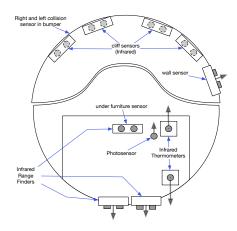


Figure 2: Sensor locations on a Katie robot.

family and gives each their unique umwelt.

We chose this minimalist design for Katie to keep all components lightweight, low power, and housed inside the cargo bay of the Create. This allows the robot to reside in a location for longer time spans and to explore (or seek shelter) under objects such as chairs, couches and tables. Retaining size and weight of the Create also maximizes mobility of the platform in physically tight or socially constrained indoor spaces, allowing the robot to do things that humans don't normally do in a space, such as getting on the ground to look under objects or to look closely at the ground and baseboards of a space.

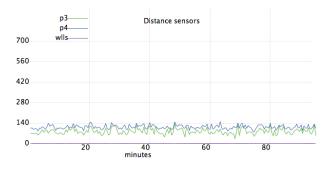
The sensorial world of Katie

The results from multiple hour tests demonstrate the resolution of the sensor data, behavior of the sensors, and the starting range of patterns that comprise the robot's umwelt. For designers, tests such as these help to decide where to place sensors and when to focus on a given data stream or not. Because there is so much complexity in even some of our simplest spaces we give the robot a basic scanning behavior, which is a 30 degree pivot every 20 seconds.

For all of our tests in this section and experiments in the next section, the following sensors are examined: two infrared (IR) range sensor aimed rearwards (integer)²; a photovoltaic cell for sensing light (integer); two IR thermometers (floats)³; and Create platform internal sensors, namely 4 integer IR sensors around the bumper to detect cliffs, a wall IR sensor, and 5 boolean wheel and bumper sensors. This results in a total of 15 sensor variables. An observation is recorded approximately once every tenth of a second.

²The Sharp GP2Y0A02YK0F has a range of 15 to 150 cm and Sharp GP2Y0A41SK0F has a range of 4 to 30cm.

³This sensor, the Melexis MLX90614, report hundredth's of a degree resolution, with an accuracy of $\pm .5^{\circ}$ C for most ranges in room temperature and $\pm .1^{\circ}$ C in human body temperature range.



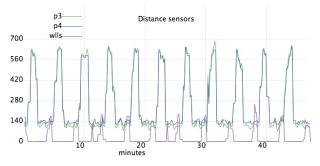


Figure 3: Infrared distance sensor data from two positions. The medium and long range sensors (p3 and p4) disagree little. In future tests the long range sensor p4 will face upward to detect when the robot is under furniture and possibly detect when in a door frame.

The robots have multiple infrared distance sensors. There are three outward facing sensors on the robot each with a different range. Figure 3 shows two 90 minute runs of data collected from a robot positioned in an open floor with no objects nearby and one positioned next to a wall. On the Create there are an additional four distance sensors that are pointed toward the ground for cliff detection. The short range on these allow for detection of small position changes if, for instance, the robot is moved or if the floor moves.

There are two infrared thermometers positioned on the robot at an upward angle of 45 degrees. Each face in opposite directions. Data from a four hour test in Figure 4 shows two cyclical patterns. The first cycle is from the rotation of robot in its scan behavior where we see a 1°F difference in the sensors at one point in each rotation.

The light sensor has a slight angle towards the front of the robot. An angle on this sensor give some direction of where light is coming into a room. Figure 5 shows data when the robot is in a sunny location in a house. Depending on the intensity of light in a specific direction, fast changes in the light can be caused by humans or animals casting a shadow on the sensor. Very fast changes, like an activation of an electric lamp, cause sharp and consistent patterns from the light sensor reading while changes in the light coming through windows caused by clouds are smooth and stable.

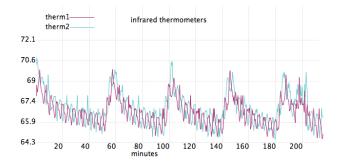


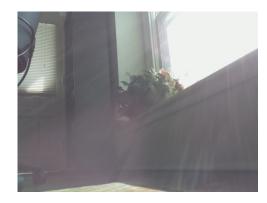
Figure 4: Infrared thermometers pick up changes as the robot faces a new direction every 20 seconds and when the furnace turns on in the house.

While the robot is stationary, the bump sensors are our the most reliable detector of presence of people and animals. The bump sensor is a large plastic bumper that covers the entire front end of the robot with a switch on each side. When an object depresses the right side the right switch is triggered. Contact with the center of the bumper triggers both sensors. All three states (contact with left side is bump=1, right side is bump=2, and center is bump=3). The wheel drop sensors are also switches. These trigger when the wheels extend all the way down (each wheel has spring to force them down when they are off the floor) are are therefore reliable for detecting when the robot is lifted.

Finally each Katie has a high resolution camera. The main purpose of the camera is to annotate the data with socially meaningful categories and to verify if the robot's classifications make sense. In the current form the camera is angled slightly upward. Robot mounted cameras can collect potentially sensitive or embarrassing information and this increases within a private space like a home. Since Americans are becoming increasingly aware and concerned about privacy we design into the robots some deference to privacy by mounting the camera with an angle toward the ground. Lowering the gaze is an embodied signal that shows deference to broader cultural concerns. The images offer another affordance that will be crucial as the robot learns more about the household and begins to move easily within it. Images can be used by people in the home to identify meaningful objects and places and such annotations can used to respond to a richer understanding of the environment.

Classifying social situations

As a first step in developing the social capabilities of the system, we would like to know if a Katie can detect differences in simple human-robot interactions. The scenarios we wish to discriminate between are: (0) an empty room, (1) someone walking across a room and (2) someone walking around the robot. Scenarios 1 and 2 are represented in figure 6. The



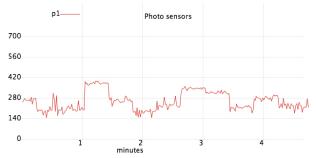


Figure 5: Light can provide some directional context if the photo sensor has a slight angle and the light source is intense. The picture was taken from the robot while it was collecting data in the graph.

robot is placed in the center of a room with measurements of distance from the robot taped to the floor. Each scenario is run at 5 different proximities: contact with the robot, 1-20 cm, 21-40 cm, 41-60 cm, and 61-80cm. Each scenarioproximity condition is run 10 times, for a total of 150 labeled experiments. For each experiment, two doors were randomly selected (by computer) from the three entrances to the room to be the starting and ending doors. During all experiments, the robot performed its scanning behavior, turning 30 degrees every 20 seconds. Certain sensors like the photo-sensor and the thermometers are correlated with time due to natural ambient variations. In order to create independence between the scenarios and the observations, the order in which the 150 experiments were conducted was random. The robot recorded an *observation*, a set of readings from its sensors, about once every 0.1 to 0.2 seconds. The mean (and standard deviation) of the number of observations per scenarios 0, 1, and 2 are: 134.92 (57.18), 38.54 (15.59) and 70.46 (21.19), respectively.

We are also interested in understanding how important various sensors are to the performance of the system, in order to avoid building new versions of the robot with useless sensors, and to help decide what new sensors should be added to the robot.

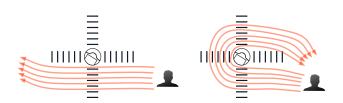


Figure 6: Scenarios 1 (left) and 2 (right).

Methods

To classify the labeled data we used three well-known classifiers implemented in the Python sckit-learn library (Pedregosa et al., 2011): random forest, boosting, and logistic regression. The first two are decision-tree classifiers, the last is a maximum-likelihood method that separates data based on linear relationships between variables. Ten-fold cross-validation was performed for each classifier as follows: the validation set of each fold contains a single, randomly-selected experiment from every scenario-proximity condition, for a total of 15 out of 150 experiments; the remaining 135 experiments (9 from each condition) comprise the training set of each fold. Sensor data is normalized by subtracting its mean and standard deviation calculated from the training set, which is especially helpful for logistic regression.

Decision tree learning is useful for classifying data with nonlinear relations. A decision tree partitions the data into regions through recursive binary splits, choosing the best predictor for the split at each step according to an impurity measure. We choose the *Gini index* for training, since it is more sensitive than misclassification error and more interpretable than cross-entropy. This impurity measure can be interpreted as the training error rate at the split. However, to evaluate the performance on validation data, we use standard *misclassification error* (Hastie et al., 2009).

The growth of decision trees is highly sensitive to noise in the data. Any errors in the first splits are propagated down to all splits below it. To reduce this variance, ensemble methods like bagging, boosting, and random forest can be used. These ensemble methods produce a forest of trees, with the final classification determined by a majority vote among them (Hastie et al., 2009). We use two such ensemble methods: *random forest*, which creates trees trained on bootstrapped data with limited access to variables at each split; and the *SAMME boosting* algorithm, which iteratively trains trees while weighting data points by their difficulty of classification, and weighting trees by their training accuracy. These methods should be able to perform well even in the presence of highly nonlinear patterns in the data, and deal well with ambiguous data points.

Finally, *logistic regression* classifies data according to linear relationships between predictors. The coefficients of this relationship are estimated by maximizing the log-likelihood

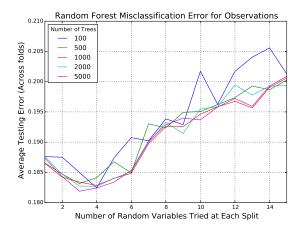


Figure 7: Mean cross-validation error of Random Forest classifier over observations, for number of trees and number of variables at each split.

of the conditional probability distribution of the class given an observation, modeled as a linear function of the variables. This classifier should do well if the data can be linearly separated, and has the additional benefit of returning a probability of class membership for future analysis. Two regularization penalties to the size of the regression coefficients are investigated: L1 (linear) and L2 (squared). These penalties subtract from the objective function of the regression, respectively, the sum of the absolute values of the coefficients and the sum of the squares of the coefficients. When the data are normalized to similar ranges, this can be used for variable selection (Hastie et al., 2009).

Results

We analyzed classifier performance for both observations and experiments. The average validation misclassification error of the random forest across all ten cross-validation folds is depicted in figures 7 and 8 for observations and experiments, respectively. Since we ultimately want the robot to classify scenario-proximity conditions, rather than single sensor observations, each experiment is classified according to a majority vote among the labels predicted for observations taken during that experiment. This also weighs each scenario-proximity condition equally, whereas per-observation error favors conditions with the most data. The performance is fairly robust to the number of trees, although more trees, as expected, tends to produce a smoother curve (more robust to changes in number of variables tried at each tree split). We can see that performance tends to be best when the classifier has access to 3 or 4 randomly selected variables at each split.

The best classifier parameters were selected according to the average validation misclassification error across all ten cross-validation folds. The corresponding errors and their

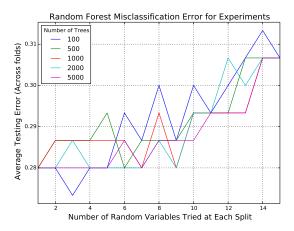


Figure 8: Mean cross-validation error of Random Forest classifier over experiments, for number of trees and number of variables at each split.

Classifier	Obs. Error	Exp. Error	
Random Forest	$0.182 \pm .041$	$0.273 \pm .094$	
SAMME Boosting	$0.204 \pm .055$	$0.293 \pm .106$	
Log. Reg.	$0.209 \pm .065$	$0.320 \pm .086$	
Trivial	$0.711 \pm .019$	$0.667 \pm .000$	
Random	$0.586 \pm .007$	$0.664 \pm .001$	

Table 1: Observation and experiment errors on 3-Scenario classification. Mean and 95% confidence interval.

95% confidence interval are shown in Table 1. The confidence intervals were calculated using the errors on each fold, assuming a t-distribution. The random forest classifier achieves observation and experiment mean error rates as low as 0.182 and 0.273, respectively. The boosting classifier achieved similar performance (figures with mean crossvalidation error not shown). The performance of the logistic regression classifiers per experiment is shown in figure 9. Performance improves with smaller regularization penalties, and for sufficiently small penalties, the performance is not significantly worse than random forest. Overall, the three classifiers can classify correctly about 80% of the time in which of the three scenarios an observation was taken, and the experiments 70% of the time.

We also computed the performance of two null-model classifiers. The *trivial* classifier labels every observation in the test set with the most frequent class label in the training set of each fold. The *random* classifier randomly labels observations in the test set with the same frequency that classes appear in the training set. The three (non-null) classifiers perform quite well, given the low-resolution sensors. They significantly outperformed the null models, although not significantly different from each other. This suggests that any nonlinear patterns in the data are not significant for perfor-

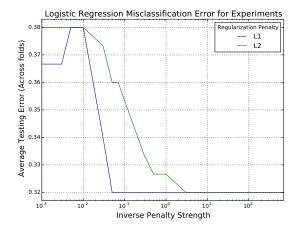


Figure 9: Mean cross-validation error of Logistic Regression classifier over experiments, for both regularization penalties

mance on this minimal social classification task—since logistic regression was not significantly worse than decision-tree classifiers.

While the performance of the classifiers is significantly better than null models, it could still be the case that most of the error would be between the two walking scenarios 1 and 2. These scenarios may be harder to distinguish, since both begin the same way. It is important that a social robot be able to determine an empty room from an occupied one. However, it is more important for social awareness that Katie can distinguish between an indifferent person and one that displays an interest in interacting, situations whose proxies here are scenarios 1 and 2 respectively.

In order to investigate whether these two scenarios are discriminated well by the same classifiers, we performed binary classifications, where scenario 2 is "positive" and scenario 1 is "negative". These values can be interpreted as answers to the question: does a person want to interact with the Katie? The average ten-fold cross-validation performance of the classifiers is presented in Tables 2 and 3 for the best parameters. Performance is reported for accuracy, balanced F1 measure (harmonic mean of precision and sensitivity) and the Matthew's correlation coefficient (MCC)⁴.

The performance of the classifiers is again quite good and significantly better than the null models, though not significantly different from one another. It is clear that the two social scenarios can be distinguished by Katie's low-resolution sensors most of the time; with accuracy reaching 90% of the time with random forest. The MCC measures the correlation between observed and predicted labels. It is zero for random prediction (as is the case of our random null model).

Classifier	Accuracy	F1	MCC
R. Forest	$0.874 \pm .077$	$0.893 \pm .079$	$0.765 \pm .118$
SAMME	$0.852 \pm .057$	$0.882 \pm .053$	$0.703 \pm .104$
Log. Reg.	$0.845 \pm .078$	$0.861 \pm .083$	$0.734 \pm .112$
Trivial	$0.647 \pm .029$	$0.785 \pm .021$	⁴
Random	$0.542 \pm .000$	$0.645 \pm .000$	$-0.001 \pm .001$

Table 2: 2-Scenario classification for observations

Classifier	Accuracy	F1	MCC
R. Forest	$0.900 \pm .075$	$0.892 \pm .094$	$0.826 \pm .128$
SAMME	$0.860 \pm .077$	$0.856 \pm .092$	$0.748 \pm .139$
Log. Reg.	$0.880 \pm .066$	$0.862 \pm .090$	$0.789 \pm .110$
Trivial	0.500 ± 0	$0.667 \pm .000$	4
Random	$0.519 \pm .001$	$0.673 \pm .001$	4

Table 3: 2-Scenario performance for experiments

For experiments, it reaches 0.826 which is a very high correlation between observation and prediction. It is also worth noticing that in the 2-scenario classification, contrary to the 3-scenario case, the performance was slightly higher for experiments than observations. This is likely due the larger number of observations gathered by scenario 0 experiments.

The relative importance of each sensor to classification performance is calculated as the expected fraction of observations that each sensor variable contributes to in the classification. This is depicted in figure 10 for the random forest classifiers. In this case, the photo sensor is the most useful, followed by the IR thermometers, the cliff sensors, and the rear-facing IR range sensors. The bumps, wheel, and wall sensor are not useful for discriminating these social scenarios. Results are similar for the boosting classifier, but with the photo sensor greatly emphasized (figure not shown).

As a proxy for variable importance, in the case of logistic regression, we can examine the average coefficients produced by the classifier for variables across folds (figure not shown). However, different sets of coefficients correspond to different scenarios. As in the case of the random forest and boosting classifiers, the photo sensor and IR thermometers are found to be important for all scenarios, but the bump sensors are also relevant in distinguishing scenario 0, since they are not activated during any scenario 0 experiment.

Future work

We see robotic development as moving from an understanding of robots as techno-scientific artifacts to sociotechnical ones. The social umwelt is woven together through technical infrastructure and we have illustrated this infrastructure with a simple robot. Furthermore, we showed that such a robot endowed with low-resolution sensors is capable of distinguishing minimal social scenarios with high accuracy. We plan on designing more sophisticated communication

⁴ The trivial classifier for observations and experiments, and the random classifier for experiments label all data into a single class, which results in a division by 0 in the calculations of the MCC

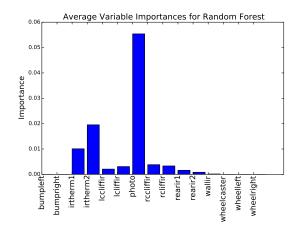


Figure 10: Random Forest variable importance

loops for humans and robots to come to a shared experience of their environment. These will extend beyond the social world of our household lab and incorporate ways for the robot to select different forms of motion. To move in this direction of more complicated social robot umwelts, we will design robots capable of interacting directly with people running common household scenarios such as arriving home from work, reading a book, watching tv, etc. We will use this information for further classification tasks and hope to refine the kinds of social states the robot can detect using simple sensors. At the same time we will have people interact with the backend of the system, looking at the world from the perspective of the robot, showing graphs and associated images, and classifying sensor data. Once the robot has a basic repertoire of meaningful classifiers it can begin making guesses about interesting/anomalous social events and then eliciting humans for annotations.

Acknowledgements

This work was supported by an Indiana University Collaborative Research Grant.

References

- Alač, M., Movellan, J., and Tanaka, F. (2011). When a robot is social: Spatial arrangements and multimodal semiotic engagement in the practice of social robotics. Social Studies of Science, 41(6):893–926.
- Almeida e Costa, F. and Rocha, L. M. (2005). Embodied and situated cognition. *Artif. Life*, 11(1-2):5–12.
- Braitenberg, V. (1984). Vehicles: Experiments in synthetic psychology. MIT Press, Cambridge, MA.
- Brooks, R. A. (2002). Flesh and machines: How robots will change us. Pantheon Books, New York.
- Clark, A. (1998). Being there: Putting brain, body, and world together again. MIT press.

- Dautenhahn, K., Ogden, B., and Quick, T. (2002). From embodied to socially embedded agents implications for interaction-aware robots. *Cognitive Systems Research*, 3:397–428.
- Dourish, P. (2001). Where the action is: The foundations of embodied interaction. MIT Press.
- Ferreira, M. I. A. and Caldas, M. G. (2013). The concept of Umwelt overlap and its application to cooperative action in multi-agent systems. *Biosemiotics*, 6(3):497–514.
- Forlizzi, J. (2007). How robotic products become social products: An ethnographic study of cleaning in the home. In ACM/IEEE International Conference on Human-Robot Interaction (HRI), volume 2, pages 129–137. ACM.
- Forlizzi, J. and DiSalvo, C. (2006). Service robots in the domestic environment: A study of the Roomba vacuum in the home. In ACM SIGCHI/SIGART Conference on Human-Robot Interaction (HRI), HRI '06, pages 258–265, New York, NY, USA. ACM.
- Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J., and Tibshirani, R. (2009). The elements of statistical learning, volume 2. Springer.
- Hoffmeyer, J. (1997). Signs of meaning in the universe. Indiana University Press.
- Kozima, H., Michalowski, M. P., and Nakagawa, C. (2009). Keepon: A playful robot for research, therapy, and entertainment. *International Journal of Social Robotics*, 1(1):3–18.
- Lee, H. R. and Sabanovic, S. (2013). Culturally variable preferences for robot design and use in South Korea, Turkey, and the United States. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 17–24.
- Matsumoto, N., Fujii, H., Goan, M., and Okada, M. (2005). Minimal design strategy for embodied communication agents. In *IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN)*, volume 14, pages 335–340.
- Mutlu, B. and Forlizzi, J. (2008). Robots in organizations: The role of workflow, social, and environmental factors in human-robot interaction. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, volume 3, pages 287–294.
- National Science Foundation (2013). National robotics initiative.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikitlearn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Shibata, T. (2012). Therapeutic seal robot as biofeedback medical device: Qualitative and quantitative evaluations of robot therapy in dementia care. *Proceedings of the IEEE*, 100(8):2527– 2538.
- Sung, J.-Y., Guo, L., Grinter, R. E., and Christensen, H. I. (2007). "My roomba is rambo": Intimate home appliances. In *International conference on Ubiquitous computing (UbiComp)*, volume 9 of *UbiComp*, pages 145–162. ACM.
- Uexkull, J. (2001). An introduction to umwelt. *Semiotica*, 134(1):107–110.

Feedback Control of Evolving Swarms

Jacob Gold, Adam Wang, Kyle I Harrington

¹DEMO Lab, Brandeis University, Waltham, MA 02453
kyleh@cs.brandeis.edu

Abstract

We examine the effects of proportional-integral control on the fitness and genetics of an evolving swarm. Introducing controllers with a set point designed to distribute birds equally across the food in a given simulation increases the rate at which the birds accumulate energy, but also increases the rate at which they expire. We find that the amount of food the birds gather is not dependent on the force of the feedback controllers but on the quality of information they transmit. The trends we observe can help to understand and improve the results of a wide variety of systems exhibiting swarm dynamics.

Introduction

Swarming and flocking behavior is ubiquitous throughout biological and physical systems of all scales. Bird flocks, schools of fish, bacteria (Munoz et al., 2007), and even chemical reactions (Sayama, 2011) are prime examples of non-equilibrium dynamical systems. Simulations to describe these phenomena were first developed by (Reynolds, 1987). Remarkably, the overall motion is governed by relatively simple rules detailing behavior of individuals and their interactions with neighbors. Motivated by both the desire to achieve greater biological control of evolving ecosystems (Holt and Hochberg, 1997; Roderick and Navajas, 2003) and recent advances in swarm robotics (Rubenstein et al., 2012), we introduce an environmental control mechanism and explore the evolutionary consequences of environmental feedback control.

In this paper, we explore the interaction between an evolving swarm and an environmental feedback controller. We consider a flock of birds with localized energy sources that provide birds with the energy necessary for their survival. In real-life systems the behaviors of agents evolve over successive generations in order to favor the most successful. Natural selection is accounted for in our model by an energy system where birds expire and are replaced by offspring of the remaining population.

We introduce control into our system in the form of local proportional-integral-derivative (PID) controllers that are capable of attracting and repelling birds. These PID controllers drive the system towards a particular state defined by the PID controller, allowing the system to be optimized. Optimization of heterogeneous swarms has several applications, such as commercial pollination with flying agents (Berman et al., 2011b,a), electric power systems (Fukuyama et al., 1999), and general optimization problems (Eberhart and Kennedy, 1995). Furthermore, recent advances in robotics have made it possible to experiment with large-scale physical swarms of robots (Rubenstein et al., 2012), allowing novel swarm control techniques to be easily tested.

The Evolution of Swarms

Since the original proposal of the Boids algorithm (Reynolds, 1987), the collective dynamics of swarms has been a common topic of study in artificial life. While many studies focus on the effects of swarming algorithms on collective dynamics, we focus specifically on the evolution of swarms. In (Spector and Klein, 2002), the evolution of simple swarming parameters is used in conjunction with a visual simulator to study emergent dynamics. This study was later extended to a radical degree with endogenously evolving computer program controllers for agents in the evolving swarm (Spector et al., 2005). Other work utilizing evolutionary algorithms in conjunction with swarming behavior has mostly been pursued in the context of particle swarm optimization, such as (Zhang and Xie, 2003). We develop our model based upon (Spector and Klein, 2002), to maintain an experimentally tractable degree of complexity.

Model

Simulation is performed in 3D with the Brevis simulator (Harrington, 2014), a scientific and artificial life simulator. Brevis provides simulation and visualization capabilities via the Java JVM and the programming language, Clojure. One particular feature of importance in swarming simulations is neighborhood detection. Brevis provides a nearest neighbor algorithm, allowing for fast lookups of operations commonly used in swarm algorithms.

The core features of the simulation are a population of

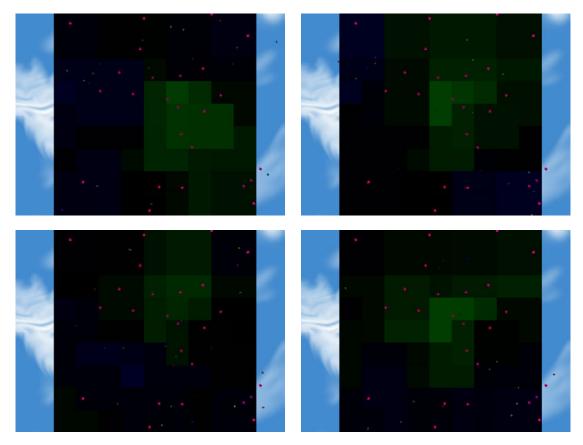


Figure 1: Example of a feedback-controlled evolving swarm and environment simulated with Brevis (Harrington, 2014). Green regions are attractive, blue are repulsive. Pink dots are food, colored cones are birds.

evolving birds, a set of energy sources, and a feedback controller (discussed later). The birds in our simulation use a simple algorithm to update their accelerations. For each bird, we consider the first bird and the first food in its list of neighbors, if any exist within the simulation-defined neighborhood radius (200 units, in this case). For both the neighboring bird and food, the direction vector from the bird to the neighboring object is multiplied by a constant, depending on whether that neighboring object is considered close to or far from the bird. This vector will be 0 if no neighbor of that type exists. The acceleration of the bird is updated to be the sum of these two vectors and passed to Brevis to update their position and velocity.

Energy

Each bird and food has an energy associated with it. A bird's energy will decrease at a constant rate (0.25 units/time), while a food's energy will increase at a constant rate (0.1 units/time). The change in energy in each iteration is proportional to the time step of that iteration. In these simulations we use a fixed time step of 1. If a bird collides with a food it gains energy at a rate of 0.005 units/time, and the food loses energy at a rate of 0.005 units/time. Also, if two

birds collide with each other, they will both lose 0.001 units of energy. When a food reaches zero energy, it will be removed from the simulation and a new food will be created at a random position. When a bird reaches zero energy, it will die and be removed from the simulation, and a new mutant bird will be created at a random position to replace it.

Evolutionary Algorithm

A genome is associated with every bird, made up of the information used to update their accelerations. Each bird has a distance associated with food and a distance associated with other birds. If a neighbor is closer than the corresponding distance, it is considered close; otherwise, it is considered far. Each bird also has four coefficients for determining acceleration with respect to neighboring objects: close food, far food, close birds, and far birds. The distance genes range over the nonnegative numbers, while the coefficients can be positive or negative. These genes are all listed below.

Genes

- neighborC: Neighbor coefficient for close behavior
- neighborF: Neighbor coefficient for far behavior

- neighborD: Neighbor distance
- foodC: Food coefficient for close behavior
- foodF: Food coefficient for far behavior
- foodD: Food distance

Every time a bird runs out of energy, its replacement will obtain a mutated copy of a genome of a bird still alive in the simulation. The mutation consists of multiplying the existing value by a random number picked from an even distribution between 0.5 and 1.5. This serves as the fitness evaluation of our simulation. Birds which are fit are able to keep themselves alive longer, and by doing so they are more likely to have their genes copied when a new bird is created.

Feedback Control

Feedback control serves a variety of purposes in engineered systems, as it allows for real-time correction. Many algorithms exist which allow the feedback controller to calculate or learn what control parameter will move the system towards the set point. We adopt a proportional-integral-derivative (PID) algorithm as our feedback controller (Astrom and Hagglund, 1995; Astrom and Murray, 2008).

Many biological systems naturally self-organize, and often utilize control techniques to achieve this organization. Insects, such as ants, bees, and termites, deposit pheromones to indirectly coordinate with their colony. This stigmergic communication aids in collective behaviors such as navigation, defense, and brood care. In previous work, we found that stigmergic communication can improve the performance of teams of heterogenous agents in real-time strategy games by communicating agent density and state information (Olsen et al., 2008). However, in the work at hand, we consider a top-down environmental feedback controller that regulates agent density.

PID Controller

Originally developed to control the heading of large ships, the PID control algorithm is now used in a variety of controllers, such as thermostats and automobile cruise controls (Minorsky, 1922). We specify the desired state of the system, or set point, and the PID controller modifies parameters of the system until it is in that state. Formally, the output of the PID as a function of time is given by

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau)d\tau + k_d \frac{de(t)}{dt}, \qquad (1)$$

where e(t) is the error, defined as the set point minus output:

$$e(t) \equiv S - u(t). \tag{2}$$

The constants k_p , k_i , and k_d simply adjust the weight of the proportional, integral and derivative terms, respectively.

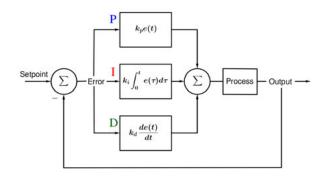


Figure 2: Illustration of the PID Controller.

The goal of the PID controller is to obtain the output such that the set point is obtained, or equivalently when the error is zero.

In our simulation we have a tile floor, and each tile is associated with a feedback controller. The input to the controller is the number of birds within a certain radius of the center of the tile. The set point $S_{\rm tile}$ of the controller in each tile specifies the number of birds which should be in that radius, which we have defined as

$$S_{\text{tile}} \equiv \frac{B_{\text{total}}}{F_{\text{total}}} F_{\text{rad}},$$
 (3)

where B represents birds and F represents food. Our simulation ensures that, $B_{\rm total}/F_{\rm total}$ is constant, so $S_{\rm tile} \propto F_{\rm rad}$. We see then that the more food within a specified radius of a PID controller, the higher the set point and thus number of desired birds in that tile will be. It is important to note here that our simulation does not use periodic boundary conditions, so birds that fly far from our tile floor will not be subject to PID control, but still expire when they reach zero energy.

For our control algorithm, we consider only the first two terms of the PID algorithm. The output of the controller as a function of time becomes

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau) d\tau.$$
 (4)

The proportional term makes a tile attractive if there are fewer birds locally than the set point, and repulsive if there are more birds than the set point. The integral term helps to overcome any steady-state error that may exist in our system due to overshoot and undershoot caused by the proportional term alone. For example, a bird outside an attractive tile's radius may circle that tile as it experiences a constant force; the integral term means the accumulated error will gradually make the tile more attractive in this situation, and the bird will be pulled inwards.

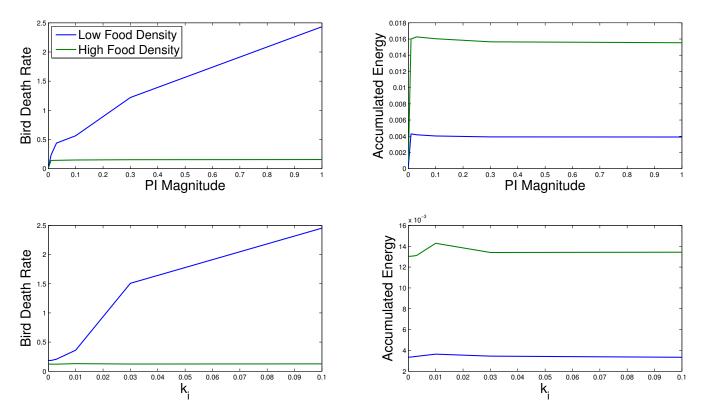


Figure 3: Average fitness for varying values of PI magnitude and k_i . Note the sharp increase in accumulated energy between PI magnitudes of 0 and 0.01

Experiments

All of the following experiments are conducted in Brevis (Harrington, 2014) for 50,000 timesteps. We sample over 3 parameters of the feedback controller: strength of feedback control, integral magnitude, and range of control. Each parameter set is tested on 10 unique random seeds.

Strength of Feedback Control

Our first set of experiments consist of varying the parameters of our PI control on simulations with low (5 food) and high (25 food) food density. Each experiment has 50 birds. Our control experiment has no feedback. When we introduce feedback, we have a parameter which we multiply the output of our PI controller by to modify the magnitude of its effect. We considered a somewhat logarithmic series of values for this parameter: 0.01, 0.03, 0.1, 0.3, and 1. The greatest of these values roughly corresponds to the maximum force felt by birds in the no-feedback experiment.

Integral Magnitude

We also consider a range of values for our integral constant k_i of 0.001, 0.003, 0.01, 0.03, and 0.1, holding the proportional constant k_p at a fixed value $k_p = 1$. At a k_i of 0.1 the accumulated error can saturate extremely quickly, so we do not consider values greater than this.

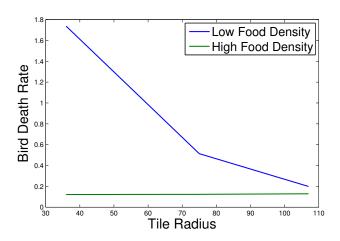
Range of Feedback Control

The final parameter we vary is the size of our feedback controller. To do this, we select three different radii of increasing size, corresponding to: the circle circumscribing the tile associated with the controller, the circle inscribed in the 3x3 neighborhood of tiles around the controller, and the circle circumscribing the neighbors in the cardinal directions.

Results

There are two primary statistics in our simulation that indicate the fitness of the birds: the rate at which they expire, and the rate at which they accumulate energy. A lower death rate and a higher energy gathering rate would correspond to fitter birds. We expect that having a feedback controller to distribute the birds evenly across the food could improve all aspects of their fitness. However, the results that we see indicate that introducing control increases the rate at which the birds accumulate energy, but it also increases the rate at which they expire.

The feedback control manages to help some birds locate and stay near the food, so they are able to collect more energy overall. This can be seen Fig. 3 with a sharp increase in accumulated energy as soon as the magnitude of PI control becomes greater than zero. As the force of the PI control continues to increase, the amount of energy gathered by the



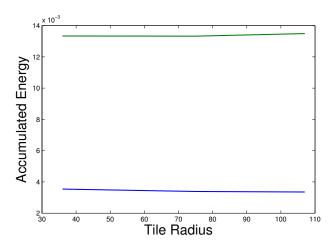


Figure 4: Average fitness for varying tile radii.

birds remains roughly constant. If we imagine a given layout of food, it is likely that there is an optimum distribution of birds across that layout. As we increase the force exerted on the birds by the feedback controller, the genes of the birds will change to compensate for the increased force in such a way that they still distribute themselves evenly across the food. Going from no control to some level of control provides additional information to the birds, allowing them to locate food more easily. Increasing the force of that control does not provide them with any new information, so it does not improve their fitness.

Varying the integral constant of the PI control informs us of what the optimum constant should be. For both low and high food density, we see a maximum of food accumulated when $k_i = 0.01$, shown in Fig. 3. When k_i is too low, the accumulated error barely provides any information compared to the proportional term about what force a controller should be exerting. When k_i is too high, the accumulated error will quickly saturate, resulting in behavior akin to a proportional term with some time lag. When k_i is at a near-optimum value, it will allow each controller to compensate for any steady-state error which may occur. While increasing the magnitude of the feedback does not improve the fitness because it does not provide additional information, the integral constant affects the quality of the information provided by the controllers, which is why it has a more noticeable impact on the amount of energy the birds accumulate.

Increasing the magnitude of the PI control does negatively impact the fitness of the birds as well. Particularly in the low food density case, it will cause the birds to expire at a higher rate. The same trend exists for increasing values of k_i , since higher integral constants will allow for a greater force to be exerted by the controllers, meaning it will have similar effects. We believe increasing the amount of feedback increases the death rate because there exists the possibility that there will be large regions of the simulation with-

out any food. Since the set point of controllers in this region will be 0, if they exert a force on the birds it will only ever be repulsive, due to accumulated error. If birds are introduced into the simulation in these large repulsive regions, it is likely that they will be pushed away from the inner part of our simulation where all the food is located. This decrease of fitness as a result of the introduction of feedback control is not one which needs to exist, but simply a result of our specific implementation. It may be avoided with a change such as only allowing the controllers to exert a positive force of varying magnitude on the birds.

The effects of the radius around a feedback controller for which a bird or food is considered nearby are shown in Fig. 4 and are rather straightforward. In the low food density case, we see a decrease in the bird death rate for increasing tile size. This agrees with our earlier analysis that large regions of repulsive tiles will cause the birds to expire at a higher rate. When the tile radius is larger, more tiles will have food nearby. This will result in more tiles which will tend to be attractive and fewer tiles which will tend to be repulsive, so fewer birds will be pushed outside the region containing the food. Tile radius does not have an effect on the rate at which the birds accumulate energy. This is likely because the birds are able to get the same amount of information from the feedback controllers regardless of their radius. If neighboring controllers overlap, the information about where the bird should go exists in the difference between their outputs. Tile radius may have more of an effect in a system where the controllers are not equidistant from one another.

Looking at the genomes of the birds for varying parameters provides us with more information about the effects of PI control. In practically every simulation the distance for considering food to be far or close converged to 0 for all birds; the ability to have two coefficients did not grant enough extra information to be necessary. This makes it

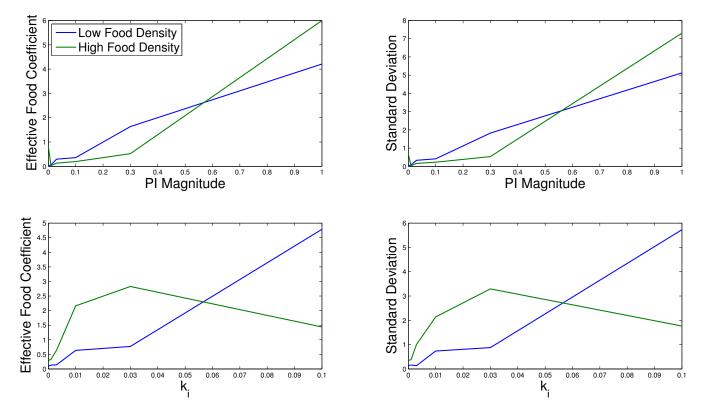


Figure 5: Effective food coefficient gene observed for varying values of PI magnitude and k_i , along with its standard deviation. The effective food coefficient is the value the birds actually use, as in nearly all simulations the food distance converged to 0.

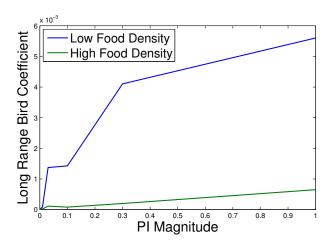
easy to analyze the food coefficient, as we can simply average across the far coefficient. For every parameter we varied we can see a trend that this food coefficient increases as that parameter increases. Since we performed mutation by multiplying the value for a gene by a random number centered at 1, simulations which converge to a greater value will have a greater standard deviation.

For increasing PI magnitude, the forces exerted by the feedback controllers will vary more rapidly with time as the birds move around and the proportional term quickly adjusts. This means that for birds to compensate to this rapidly changing environment, they must be able to accelerate rapidly. This is why we see the positive relationship between PI magnitude and food coefficient in Fig. 5. For increasing values of k_i , we see two different trends for low and high food density. Increasing the integral constant will increase the magnitude of the feedback controller's output. In the low food density simulations the large regions which become repulsive do so because of accumulated error, so for high k_i the birds need to accelerate quickly towards the food to cross these regions. In the high food density case k_i will similarly increase the overall output of the feedback, but the birds do not have to deal with large repulsive regions. For lower values of k_i we still see a positive trend as the accumulating error will result in changes that the birds need to react to, albeit more slowly than the proportional term. However, for higher values of k_i the accumulated error will quickly become saturated, resulting in what will often be a constant attractive or repulsive force. As the strength of an attractive force of this nature will be large relative to the other forces on a bird, that bird will be more successful if it allows itself to be pulled toward the food by the control instead of accelerating towards it on its own and overshooting the target.

The genes for neighbor coefficients display a similar behavior to that of the food coefficient, though the magnitude of these values is considerably lower, as seen in Fig. 6. As all the other birds will be attempting to find and stay close to food, moving towards them will often result in moving towards the food they are near. Since increasing the magnitude of control increases the food coefficient, it follows that it will increase the bird coefficient as well.

Conclusions

We have introduced an environmental feedback controller to a model of swarm evolution. This introduction of control has a significant effect on simulation dynamics; however, the benefit of control is clearly dependent upon the goal of the evolving swarm. In this case, control does not reduce the rate at which birds expire, but instead leads to an increase in energy accumulation. This type of behavior is desirable in a



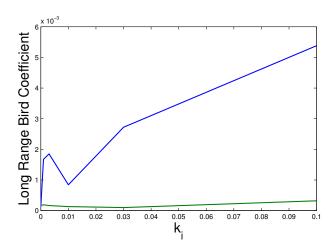


Figure 6: The coefficient birds to determine their acceleration with respect to other birds. Since the bird distance gene did not converge to 0, these values have been averaged over the coefficient birds would use at distance greater than 50, which is higher than the average value of bird distance in any simulation.

number of situations, for example search and rescue, reconnaissance, and crop pollination. Our model can be readily extended to test other ideas relating to the control of evolving swarms, and opens new possibilities for studying the interface between control theory and artificial life.

Future Work

In previous work we have introduced the use of gene regulatory networks to actively tune the parameters of a reinforcement learning-based control algorithm, where the tuning allows the behavior of controllers to change over time (Harrington et al., 2013). Introducing this additional layer to the controller may allow controllers to adopt different modes of control based on context.

To further explore the effects of a PI controller, we plan to incorporate energy sinks in the form of stationary obstructions that take away energy upon collision. Currently, the average bird death rate increases as the magnitude of the controller increases. We expect with the addition of sinks that their associated PI controllers could repel the birds away, and thus decrease the death rate.

We also plan to allow the PI controller to regulate motion in all three dimensions, rather than only the horizontal plane. It would then be possible to place PI controllers on the sources and sinks of energy themselves. The function of the PI controller could also be modified, with the goal of sustainability in mind, pushing birds more strongly towards food sources with high energy as opposed to the nearest food source.

In contrast to our PI controller that promotes convergence toward the food, it would be interesting to implement a pesticide that aims to repel birds from energy sources. We could examine the genes that characterize fit birds in such a system, and if a particular combination of genes can overcome the pesticide. Furthermore, we could construct an optimization problem of how to apply pesticide by introducing a cost proportional to the pesticide strength. We would then search for the minimum pesticide strength that successfully repels birds.

We believe our PI control mechanism which functions on an input of how evenly the birds are distributed could have applications for particle swarm optimization (Zhang and Xie, 2003). Using feedback to try to distribute the particles evenly through the search space could help to maintain diversity while still utilizing the benefits of swarm behavior.

Acknowledgements

KIH is funded by the Brandeis University School of Computer Science. Computing support was provided by the Brandeis University HPC cluster. We thank Seth Fraden, and Jordan Pollack for advice and support. We also thank the UMass BINDS lab for comments, in particular Patrick Taylor.

References

Åström, K.J., and Hägglund, T. (1995). *PID Controllers* (2nd ed.). Research Triangle Park, N.C.: International Society for Measurement and Control.

Åström, K. J., and Murray, R.M. (2008). *Feedback Systems:* An Introduction for Scientists and Engineers. Princeton: Princeton University Press.

Berman, S., Kumar, V., and Nagpal, R. (2011a). Design of control policies for spatially inhomogeneous robot swarms with application to commercial pollination. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 378–385.

- Berman, S., Nagpal, R., and Halász, A. (2011b). Optimization of stochastic strategies for spatially inhomogeneous robot swarms: A case study in commercial pollination. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3923–3930.
- Eberhart, R. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Micro Machine and Human Science*, 1995. MHS'95., Proceedings of the Sixth International Symposium on, pages 39–43.
- Fukuyama, Y., Takayama, S., Nakanishi, Y., and Yoshida, H. (1999). A Particle Swarm Optimization for Reactive Power and Voltage Control in Electric Power Systems. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1523–1528, Orlando, Florida, USA. Morgan Kaufmann.
- Harrington, K. I. (2014). Brevis (Version 0.7.17).
- Harrington, K. I., Awa, E., Cussat-Blanc, S., and Pollack, J. (2013). Robot Coverage Control by Evolved Neuromodulation. In *IJCNN 2013*, page 543-550.
- Holt, R. and Hochberg, M. (1997). When is biological control evolutionarily stable (or is it)? *Ecology*, 78:1673-1683.
- Minorsky, N. (1922). Directional Stability of Automatically Steered Bodies. *Journal of the American Society for Naval Engineers*, 42(2):280-309.
- Munoz, M., Lopez, J., and Caicedo, E. (2007). Bacteria swarm foraging optimization for dynamical resource allocation in a multizone temperature experimentation platform. In *Analysis and Design of Intelligent Systems using Soft Computing Techniques*, pages 427–435.
- Olsen, M., Harrington, K., and Siegelmann, H. (2008). Emotions for Strategic Real-Time Systems. In AAAI Emotion, Personality, and Social Behavior Technical Report, pages 104–110. Março.
- Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH Computer Graphics*, volume 21, pages 25–34. ACM Press.
- Roderick, G. and Navajas, M. (2003). Genes in new environments: genetics and evolution in biological control. *Nature Reviews Genetics*, 4:889-899.
- Rubenstein, M., Ahler, C., and Nagpal, R. (2012). Kilobot: A low cost scalable robot system for collective behaviors. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3293–3298.

- Sayama, H. (2011). Seeking Open-Ended Evolution in Swarm Chemistry. In *Artificial Life (ALIFE)*, 2011 *IEEE Symposium on*, pages 186–193.
- Spector, L. and Klein, J. (2002). Evolutionary dynamics discovered via visualization in the breve simulation environment. In *Workshop Proceedings of the 8th International Conference on the Simulation and Synthesis of Living Systems*, pages 163–170.
- Spector, L., Klein, J., Perry, C., and Feinstein, M. (2005). Emergence of Collective Behavior in Evolving Populations of Flying Agents. *Genetic Programming and Evolvable Machines*, 6(1):111–125.
- Toner, J. and Tu, Y. (1998). Flocks, herds, and schools: A quantitative theory of flocking. *Physical review E*, 58(4):4828–4858.
- Zhang, W.J. and Xie, X.F. (2003). DEPSO: Hybrid Particle Swarm with Differential Evolution Operator. *IEEE International Conference on Systems Man and Cybernetics*, pages 3816–3821.

Computation and Scientific Discovery? A Bio-inspired Approach

Ioan Muntean¹

¹ The *Reilly Center for Science, Technology and Values*, University of Notre Dame, Notre Dame, IN 46556 imuntean@nd.edu

Abstract

Philosophers argue that scientific discovery is far from being a rule-following procedure with a general logic: More likely it incorporates creativity and autonomy of the scientist, and probably luck. Others think that discovery can be automatized by some computational process. Based on a concrete example of Schmidt and Lipson Schmidt and Lipson (2009), I argue that the bottom-up discovery is computable and that both aspects of creativity and autonomy can be incorporated. The bio-inspired evolutionary computation (genetic algorithms) are the most promising tool in this respect. The paper tackles the epistemology of applying a evolutionary computational and genetic algorithms, to the process of discovering laws of nature, invariants or symmetries from collections of data. Here i focus on more general aspects of the epistemology of evolutionary computation when applied to knowledge discovery. These two topics: computational techniques applied in science and scientific discovery taken separately are both controversial enough to raise suspicions in philosophy of science. The majority of philosophers of science would look with a jaundiced eye to both and ask whether there is anything new to say about discovery and computers in science. This paper is a first stab to the philosophical richness of computational techniques applied to the context of discovery. I discuss the prospect of using this type of computation to discover laws of nature, invariants or symmetries and appraise their role in future scientific discoveries.

Is scientific discovery an algorithmic process?

I argue in this paper for a deeper connection between bioinspired computation and the process of scientific discovery. Based on new concrete results of Schmidt and Lipson 2009, I infer here some epistemological consequences for using evolutionary computation in scientific discovery.

Knowledge is central to virtually all advanced forms of life; discovery and learning characterize us as a species as well as other higher order animals. We discover in order to survive and adapt. Science is just another specific form of knowledge in which data and experiments play a fundamental role in conjecturing hypotheses about the world. If discovery is probably intrinsically linked to our evolution as a whole, scientific discovery played a central role only in the evolution of humanity in the last four centuries or so (a good turning point is the work of Francis Bacon and its influence during the "Scientific Revolution").

How do we infer laws and generalizations from data? How do we discover new models and theories? Are creativity and autonomy of scientists major cognitive faculties that define and shape science, or, on the contrary, is scientific discovery just a process of following rules, methods and algorithms? The nature of scientific discovery, together with, arguably, artistic creativity, moral decision making and religious experience are among those faculties that define us as humans better than anything else.

These fundamental questions about the nature of scientific discovery are germane to the discussion of artificial scientific discovery. As I link the process of discovery to human life as a species, it is germane to investigate philosophically the paths to an artificial process of scientific discovery. Can we create machines that would perform activities deemed by many as "human-only"?

The broader scope of this paper is to investigate the possibility of a cooperation between the human scientist and the artificial discoverer. I based my argument on a specific

Two approaches to scientific discovery

For the purpose of this paper, the scientific endeavor can be divided between the context of discovery and the context of justification. The distinction can be traced to H. Reichenbach's early works but it is very clearly expressed in Reichenbach (1949). After introducing the infamous distinction, Reichenbach discussed the reliability of a logic and epistemology of discovery. Epistemology is a rational reconstruction of a thought process. In a common interpretation, there is no epistemology of discovery, which is basically a subjective and irrational process: P. Duhem, E. Mach, K. Popper, R. Carnap, C. Hempel, or R. Brainwaite for different reasons deemed discovery as irrelevant when compared to the context of justification. The iconoclastic view of scientific discovery as a "happy guess" or "mystic presentiment" is discussed in Koestler (1959). In a different key, M. Curd and Th. Nickles interpreted Reichenbach's discovery-justification distinction as not excluding an epistemology of discovery. There is an epistemology of discovery, with or without a logic of discovery. So epistemology is much a broader area than logic in this specific framework.

For both these contexts it is relevant to ask this question: is science based on deductive logic, induction or on heuristics? A similar question can be asked about the nature of discovery: is scientific discovery algorithmic, nearly algorithmic or, on

the contrary, is it non-discursive, not re-constructible, non-reproducible, singular, a "Eureka"-like mental episode? Is discovery merely a psychological process with no epistemological significance (when compared to the process of justification, for example)?

There are perhaps two main programs in the philosophy of scientific discovery. First, there is a strong program aiming to formulate a general logic for scientific discovery, to encompass all scientific discoveries under one formalism Simon (1973); Hanson (1958). The connection proposed by Langley, Simon, Bradshaw and Zytkow (Langley et al., 1987) between discovery and the heuristic search procedure falls under this strong program. But this strong program felt in disgrace for several reasons and was replaced with a weaker program that gives up the idea of a formal and general logic of scientific discovery and tackles the epistemological aspects of particular discoveries Nickles (1980b,a); Meheus and Nickles (2009). Here epistemology can be both descriptive and normative and more attention is paid to non-formal and non-logical epistemological aspects of discovery: heuristics, search, risky generalizations, etc. This weak program is more sensitive to the specific conditions of the discovery and of the specific nature of the discoverer. One can ask two questions:

- (1) How do individual scientists, with their limited cognitive faculty, discover new scientific theories? By following a set of rules or by sheer creativity?
- (2) How new theories can be discovered by scientists aided by computers, by Artificial Intelligence systems, or any system other than individual scientists?

The descriptive epistemology of scientific discovery can answer (1) by a careful analysis carried within history of science. Here the discoverer is an individual—the lone genius of Kant, or any scientist experiencing the "Eureka" moment of discovery. We face here a "dilemma of explanation" if we have a theory about scientific discovery as algorithmic Nickles (1980b); Wartofsky (1980):

(3) DILEMMA OF ALGORITHMIC EXPLANATION: The dilemma is then: either the theory succeeds, and the concept of discovery is explained away, or reductively eliminated—or the theory fails, and discovery remains unexplained.

I emphasize here the novelty of question (2). First, it does not have a complete answer in the history of science, because the computer-aided scientific discovery or discoveries made by large teams of scientists have a shorter history—when compared to scientific discoveries made by individuals. When the discoverer is a collaborative team, a whole scientific communities, a team working with computers, or a set of computational processes, or all these working together, rationality and creativity may well have radically opposite meanings. The answer to (1) does not entail an answer to (2), and vice-versa. Communities, computers or other entities may discover scientific laws, patterns, or theories by an altogether different mechanism than human scientists do, with or without explaining away creativity.

This paper aims to answer (2) and show in what sense there is "a third way" in Wartofsky's dilemma (3). The way in which

computers and artificial intelligence are used in science may elucidate the normative part of this epistemological approach, but we do not need to equate computational techniques with rational agents, machines, number crunching devices, etc. I do not identify rationality with logic, irrationality with creativity, or machines with logic and creativity with humans only. When used in the scientific discovery, the computational technique comprehends several elements such as: creativity, rule-following procedures, logic etc. I think there is something interesting for philosophers to study about discovery and about computation, taken separately or when computation is directly applied to scientific discovery.

The skeptic against computers used in areas in which human knowledge reigns may raise important questions: Are current computational techniques versatile enough to reproduce, and eventually enhance, the process of scientific discovery? If so, which type of computation is the most promising? And moreover, is this process going to slowly replace humans with machines, even in the process of discover? I reckon that all these questions are attractive from a philosophy of science point of view. It is even more contentious whether a computational process can discover solutions to problems that humans (alone) cannot discover.

In focusing on the epistemology of scientific discovery and the possibility of its algorithmic reconstruction, the current approach is more local and partial: I focus on a specific bottom-up approach to discovery: inferring invariants and laws of nature from large sets of data, and on a specific type of computation: the evolutionary computation implemented by genetic algorithms.

The philosophy of computation in science follows the debates on the relation between data, phenomena, models and theories. For the purpose of my analysis, two contexts of computational science are relevant, both inspired by recent discussions on applying science/applied science Morrison (2006); Bod (2006); Boon (2006). (a) The computational technique starts from a scientific theory and move towards the data: here computation is the application of a theory or a "top-down" approach. Or (b), computation is a heuristic tool that starts from data and builds a theory in a "bottom-up" approach. Each of these two approaches may have their own specific computational turns: computational techniques used in one may or may not be as revolutionary as they seem in the other. Differentiating these two contexts may help the philosopher argue for the novelty of the epistemological aspects of (b) when compared to (a).

Evolutionary Computation and the Bottom-up Approach to Theory-building

On different occasions, philosophers and scientists alike pointed out to a major difference among two types of scientific reasoning (Th. Kuhn, L. Laudan, among others). On one hand, one has the rule-based reasoning in which new theories or models are inferred from a set of rules. The system of abstract rules is used to solve problems. The rules in general are content-neutral and in the ideal situation they can be applied to virtually any new set of data. On the other hand, one witnesses case-based reasoning in science. Th. Kuhn and K. Popper asked incessantly: is science applied by following rules? Exemplars are solutions to previous problems that scientist learn during their scientific education and solve future puzzles based on an "acquired similarity" Kuhn (1962).

¹For reasons why the strong program failed, see Curd (1980); Laudan (1980).

Scientists try to make a new phenomenon fit to one or more previous phenomena.

A relevant step forward is to show that neither science, nor computation can be reduced to a succession of rule-following procedures. If we restrict computers to rule-following, then there is little chance, if any, that computational techniques can be useful in scientific discovery. Some philosophers of science have analyzed computation as heuristics device in the discovery of new theories. Here concrete results are less notable than in (a). Computer scientists try to use algorithms to discover laws of nature, invariants or patterns in data at least since the 1970s: the most known are the packages DENTRAL, EURISKO, GLAUBER, STAHL and BACON Simon et al. (1981); Mitchell (1997); Waltz and Buchanan (2009). They are designed for a theory-building procedure, when the scientists have little or no idea about how the theory is supposed to look like Keller (2003); Galison (1996); Langley (1979); Barberousse et al. (2007); Pennock (2000, 2007). There is a similarity between the Case-Based reasoning suggested by Kuhn and similar AI techniques used in problemsolving Bod (2006). A case-based procedure always retrieves cases whose problem is similar to the problem being solved. The procedure discussed is data-oriented as opposed to rulebased processing. Computers mimic frequently the process of learning, which is not completely based on rules. According to Bod, data-oriented procedures in computers are similar to the way scientists explain new phenomena "by maximizing derivational similarity between the new phenomenon and previously derived phenomena" Bod (2006).

Therefore, neither scientists nor computers follow strict rules, but reuse previous results in order to solve new problems. For Bod, previous patterns of derivations are learned and accumulated, not phenomena in themselves. Rules are always present, but they are complemented with corrections, normalizations, exemplars derivations, adjustments, all stored and reused from previous cases. In context (b), in the dataoriented discovery process, something else is needed than rule-following procesures. This takes us a step towards answering (1) and solving dilemma (3). As P. Langley et al., P. Thagard (1998) and L. Darden (1998) have argued, bringing in computation into the discussion on scientific discovery should majorly boost philosopher's interest in discovery. But, as my argument goes, the nature of computation plays a central role in dismissing (3) as a false dilemma and answering (2). I show that once we move to a new type of computation, (3) is based on some false assumptions if we give up the very restrictive concept of algorithm and adopt a general concept of computation.

Based on the concrete case study (Schmidt and Lipson, 2009), I show in what sense creativity and rationality can in fact go hand in hand in the case of genetic algorithms applied to scientific discovery. The answer lies in the artificial life metaphor used by Schmidt and Lipson. Computational results in this context are still rare, but as my argument goes, this case cuts deeper into the computational epistemology. More concretely, in the following two sections I address these questions:

- (4) What are the epistemological consequences of using evolutionary computation in scientific discovery?
- (5) Is evolutionary computation the appropriate type of computation for the process of discovery?

Evolutionary Computation

Roughly speaking, computer algorithms were born based on three distinct analogies: algorithms as "formal proofs", algorithms as "learning processes" and algorithms as "searching procedures for optimality". The latter inspired the area of evolutionary computation, as the paradigm for optimality is an organism optimally adapted to its environment.

How is "search" related to "life"? In the 1930s, S. Wright (1932) interpreted a biological species as a system that evolves in time by exploring a multi-peaked landscape heuristic of optimal solutions to a "fitness problem". The operation of optimization of search which is typically performed by an algorithm can mimic a living organism that over a long period of evolution fits the environment. On the other hand the process of adaptation and evolution is not smooth.

Organisms are subjected to *random* mutations, too. Taken the biomimetic strategy on step forward: Is it a good idea to add randomness to algorithms? There are several types of *stochastic* algorithms each of them being more or less *biomimetic* in their nature. Biomimetic strategies are widely used in robotics and artificial intelligence, but they are almost ignored by philosophers.² Are they useful when applied to scientific discovery?

After a serendipitous proposal by A. Turing in the early 1950s, Evolutionary Computation (*EC*) was rediscovered and reinvented at least ten times before the 1980s (Fogel, 1998). The milestone is J. Holland's work (1975). Following Turing and von Neumann, Holland was able to see the potential of using the knowledge on natural adaptation process to improving search techniques and applied the principles of natural selection directly to problem-solving algorithms. One fundamental difference, not available in Turing's time, is that selection occurs better at the level of population, not at the level of individuals.

The elements of a genetic algorithm

Genetic algorithms are iterative procedures of *searching* for the optimal solution to a problem P. They are based on the metaphor of biological processes in which organisms: (a) *non-consciously* adapt to the "environment" P and (b) are selected by a *supraindividual* mechanism such as selection.³ The question is whether we can generate algorithms in the same way organisms are created through evolution.

Genetic algorithms start from a given number of initial individuals randomly distributed in a given space, called the initial population. The genetic algorithm transforms individuals, each with an associated value of fitness, into a new generation by using the principles of survival-of-the-fittest, reproduction of the fittest and sexual recombination and mutation. Similar to Wright's landscape, the genetic algorithm finds "the most suitable" or the "best so far" solution to the problem by breeding individuals over a number of generations.

The procedure can be stopped by a termination condition: when the sought-for level of optimality is reached or when all

²On the concept of biomimetics, see Srensen (2004); Muntean and Wright (2007).

³I take here algorithms as abstract, mathematical objects, whereas programs as their concrete instantiation on a machine. A sensitive difference is between genetic algorithms, genetic programming and genetic strategies. See Jong (2006).

the solutions converge to one candidate. The fitness function estimates the fitness to breeding of individuals in accordance with the principle of survival and reproduction of the fittest:

- Better individuals are more likely to be selected than inferior individuals.
- · Reselection is allowed.
- · Selection is stochastic.

The genetic algorithm ends with a *termination condition* that can be the satisfying of a success predicate or completing a maximum number of steps. The success predicate depends on the user's choice and can be deemed as a pragmatic criterion. The winner is designated at the "best-so-far" individual as the result of the run.

Here is an abstract implementation of a genetic algorithm:

- [1] produce an initial population of individuals
 - [2] WHILE `termination' not met do [3] evaluate the fitness of all individuals
 - [4] select fitter individuals for reproduction
 - [5] produce new individuals
 - [6] generate a new population by inserting some new good individuals and by discarding some 'bad' individuals
 - [7] mutate some individuals
 - [8] ENDWHILE
- [9] Call the individual(s) which satisfy
 the 'termination' condition
 the 'best-fit-so-far''

Case study: (Schmidt and Lipson, 2009): Distilling laws and invariants

To show that "algorithmic explanation" and "creativity" are *not* mutually exclusive in (3), I use as an example of computation applied directly to science the result reported in *Nature* (Schmidt and Lipson, 2009). M. Schmidt and H. Lipson have showed how symbolic regression based on evolutionary programming can be used in discovering *natural*, *non-trivial* and *meaningful* invariants in physics. Their algorithm searches over the infinite possible ways of modeling data to find the best and most useful expression available given (i) a set of data; (ii) a termination condition and (iii) a set of evolutionary path. It starts with a set of individuals which can be equations, models and scientific heuristic methods of search—not necessary mathematical objects. Each individual is tested against a bank of experimental data. Many individuals do not make

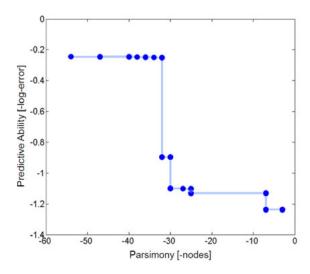


Figure 1: The Pareto front with two "cliffs".(Schmidt and Lipson, 2009, supplementary online materials)

sense mathematically or do not meet some consistency criteria, so they are discharged. Some may fit the data better than others. The software saves these individuals for "breeding", cross-combining a 'father' with a 'mother'. It is claimed that over hundreds of thousands of generations, some extremely fit individuals emerge.

Schmidt and Lipson approached scientific discovery as being data-driven. They started from a set of measured, uninterpreted set of data representing the position, velocity and acceleration of a lab experiment or a virtual system (generated by another algorithm). The method used, the "symbolic regression", is not new at all, but here the program searches for both the form and the parameters of an equation that model a given set of experimental data. They have discovered not only analytic functions from empirical data, but structures which are highly relevant to physical sciences: Hamiltonians, Lagrangians, laws of conservation, symmetries, and other invariants.

Schmidt and Lipson adopted the balance between two objectives: the predictive power and the complexity/parsimony of each candidate. By calculating the "Pareto front" of the dependence predictive ability versus parsimony, Schmidt& Lipson found that there are two cliffs where predictive ability jumps rapidly at some relatively small increase in complexity.

The Epistemology of Discovery with Evolutionary Algorithms: Risks and Advantages

One knee-jerk reaction to applying computation to science is: what is so philosophical about (yet) another tool used by scientists? Although we are nowhere near an "end of computation", the philosopher would not directly infer from its success, its epistemological relevance. Many scientific tools are successful in science, but philosophically inept, and *vice versa*. Although not yet successful, I claim that this case study is worth of a philosophical scrutiny as it sheds some

⁴The package is EUREQA, a software based on evolutionary algorithms Lab (2009).

light on some concepts such as: creativity, rule-following, knowledge production, etc. The procedure addresses some very general epistemological issues of scientific discovery. The knowledge-production in this case study uncovers interesting aspects of the scientific discovery. I frame the following epistemic "aspects" both as problems and as novel features of the scientific discovery based on evolutionary algorithms. The direct application of evolutionary computation to scientific discovery shows how productive bio-inspired algorithms can be. The most attractive feature of evolutionary computation is its ability to "explore" the logical space of solutions, even those which remains unconceived to the mind of the scientist. But the whole process is not totally automatized and the algorithm is not fully autonomous. The human scientist imposes her own meta-rules on the algorithm. On the other hand, because every solution is a model better or worse adapted to data, the bio-mimetic aspect of this example is clear: scientific models adapt to the data and create populations of solutions such that each individual contributes to the adaptation function of the population. After running the algorithm as suggested by Schmidt and Lipson, the scientist is able to explore the "tip of the iceberg", i.e. the best adapted in so far individual from a multitude of previous generations of solutions. The unconceived alternative models, although not directly present in the final solution did influence it if they were part of the intermediate generations of solutions. I relay the epistemological aspects of the genetic algorithms used in scientific discovery to the various aspects of artificial life. A stronger connection, not endorsed here, would connect knowledge in general to evolution, the are being the evolutionary epistemology. The main part of my argument is that the face of scientific discovery "as we know it" may change radically once evolutionary computation is involved in the process of discovery. I list here several aspects of this "upward epistemology" that is still nascent but very enticing philosophically.

Stochasticity versus scrutability of solutions

Genetic algorithms can be stochastic or not, depending on the mutation operator occurring in step (7) or by selecting the individuals for reproduction in step (5) (in Table 1). An algorithm becomes deterministic if exactly one parent is *identically* reproduced or if two parents are combined without adding or losing information based solely on their fitness. Genetic algorithms are stochastic in two major respects: both the operation of selection and reproduction are random. That means the results (offspring) are not direct results of the input data (the parents).

The crossover operator takes two individuals, the parents, and produces two new individuals, the offspring, by swapping substrings of the parents. Randomly choosing two parents to mate or randomly deleting or adding information from the parents will make the algorithm stochastic. Mutation is a background redistribution of strings to prevent premature convergence to *local* optima.

Weak individuals may survive "by luck" and fit individuals may not be drawn to reproduce. The advantage of a random mutation is that at least some populations, ideally a few only, could escape the traps which deterministic methods may be captured by, and end up with an unexpected and novel result. For very complex problems, this biomimetic procedure can output results which are definitely not accessible to deterministic algorithms if a delicate balance between the mechanism

of selection that decrease variation and those that increase variation (mutation) has been achieved.

Because the scientist can control this mutation operator and its frequency, the output of such a discovery algorithm is not traceable by humans. At the limit, the solution of such an algorithm may be inscrutable to humans. It is also the case that for any run, because of the stochastic element, the best individuals are not guaranteed to be selected, and the worst are not eliminated. One can say that the algorithm favors the best and marginalizes the unfit. The selection is not entirely "greedy" in the search space. We do not need to associate creativity to such a random process. As I show before, human element is not totally eliminated in this case. The creativity is blind in this case, similar to mutation in biological populations.

Rules, laws and metarules

The evolutionary algorithms do not follow a set of rules in respect of the discovery of new laws or invariants. As the case study suggests, the process of discovery is here ruled by the metarules of evolution as well as the method used to decide about the fitness function and the termination condition.

For simple laws and invariants, genetic algorithms are easily outrun but Turing machines. But given the complexity of current science, deterministic algorithms may well be worn out as aiding tools to optimality. Although this may sound speculative, let us assume that science evolved toward increasingly complex representations. Maybe the good-old-days of simple, beautiful laws of nature are gone. What if were not going to encounter beautiful laws such as:

$$F = ma; F = k \frac{m_1 m_2}{r^2}; E = mc^2; R_{ij} - \frac{1}{2}g_{ij}R = 8\pi G T_{ij}$$

anywhere down the road? For the time being, weve been lucky enough that our best laws of nature could have been fit on a "T-shirt", as it were. How do we discover more and more complex laws of nature? We are limited by conceivability and our limited resources to recognize patterns and regularities may become overtaken by the increasingly complex set of data. Time in which we could deduct laws from phenomena without any epistemic extenders may be over. More and more complex data are collected. Cosmologists, neuroscientists, sociologists, political scientists do not have the luxury to infer their laws from laws as simple as Newton's or Einstein's. What if, from now on, the would-be laws of nature wont fit even a football banner? We need to brace up for more and more complex scientific representations...

Social science, biology, suggest that we may want to drop completely the ideal of laws of nature in their simplest and purest form. In some historical cases, pre-existing theories and the accompanying mathematics were not "already there" when a major discovery in science occurred: contrast this with the received view on the "unreasonable effectiveness of mathematics". We may even need to reconsider the concept of universal laws of nature, existing independent of the way we collect and simulate data.

Now, here is a brighter perspective. Even if the good old days are bygone, there are new ways of coping with increasing complexity in the form of invariants, regularities, laws of nature and alike. Distributive knowledge in science is a tempting idea. Science made by communities of scientists, labs, research programs may steadily replace science

made by individuals. The other possible path suggested by Humphreys is a collaborative work between computers and humans. Maybe we have to face the fact that science is getting closer to the limits of our knowledge, we as limited individual brains. Philosophically put, science is getting closer to the conceivability limit of possibilities.

Triviality versus meaningless

In Schmidt and Lipson's approach, there is a problem of triviality and meaninglessness of solutions. For almost any set of empirical data there are uncountable invariants or conserved quantities, some of them being trivial, some being meaningless. The main task in this case is to find a non-trivial invariant of the system that also can be interpreted as having a meaning. Schmidt and Lipson proposed a criterion based on decomposability: the candidate equations should predict connections between dynamics of the *subcomponents* of the system. This is done by pairing the variables and looking for natural behaviors of parts of the system. More precisely, the conservation equations should be able to predict connections among derivatives of groups of variables over time, relations that we can also readily calculate from new experimental data. Ultimately, their procedure was able to infer the optimal form of the double pendulum Hamiltonian by avoiding trivial and meaningless solutions. Schmidt and Lipson included a human decision maker in their algorithm who stops the search process at certain time and imposes the constraints of the symbolic regression such as: "naturalness", "interestingness" or "meaningfulness".

Interpretation versus understanding

Bootstrapping can also be used to infer laws for more complex systems. Results about simpler systems can be used to infer equations for more complex systems. From a statistical analysis, Schmidt and Lipson inferred that terms that are frequently used and are more complex have also *meaning*. For example, trigonometric terms represent potential energy, squared velocities are associated to kinetic energy. The main claim of Schmidt and Lipson is that these terms are ready for a human interpretation:

These terms may make up an 'emergent alphabet' for describing a range of systems, which could accelerate their modeling and simplify their conceptual understanding. [...] The concise analytical expressions that we found are amenable to human interpretation and help to reveal the physics underlying the observed phenomenon. Many applications exist for this approach, in fields ranging from systems biology to cosmology, where theoretical gaps exist despite abundance in data.

Might this process diminish the role of future scientists? Quite the contrary: Scientists may use processes such as this to help focus on interesting phenomena more rapidly and to interpret their meaning Schmidt and Lipson (2009).

The outcome of such an algorithm can help *in the future* with understanding scientific results which are not strictly speaking discovered by humans. The operation of distilling laws from data does more than generating symbols, be them complex expressions of conserved quantities or equations. Similar to numerical simulations, "the results are not automatically reliable" and more effort and human expertise is needed

to decide what results are reliable and which are not (Winsberg, 2009). But in this case the computation is more than a tool or a technique because it makes the results intelligible to the human scientist and the question whether the method can be truly creative is up for grabs.

Path dependency versus global solutions

Genetic algorithms compensate some of their drawbacks by their effectiveness in global search. Remember that they maintain a population of solutions which are constantly updated with fitter new individuals and hence avoid local optima. For a certain complexity of the search space, a genetic algorithm has a better chance to find the global optimum. This changes radically the epistemological aspects of genetic algorithms. They are very efficient in solving "hard problems" where little or nothing is known about the soughtfor structure and when discovering new structures trumps the process of evaluating existing knowledge.

The case study underscores well this problem of any evolutionary computation: its path dependence. Even the nontrivial and meaningful solutions are not unique! The procedure does not produce a single set of solutions, but a set of candidates for the analytical solutions. It is known that any complex problem has a number of local maxima in the landscape of solutions with different fitness values. At different runs of the simulations, different populations can converge to different maxima. The human discoverer will always reach only one solution, whereas a set of genetic algorithms running on the same initial population will end up with different optimal solutions. This is a direct consequence of the fact that similar to biological evolution, the process is nondeterministic. As it was recently argued, this leads to a nonmodular functionality of the algorithms and hence to a limited understanding of the operations (Kuorikoski and Pyhnen, 2013). The only aspect which is etymologically accessible to the scientist is comparing results and deciding the best fit. But the way we achieved that results is inscrutable to the scientist. Previous generations and the evolution itself is in many cases too complicated to follow or alternatively, too stochastic to constitute a justification per se. As we cannot trace the proof of the algorithm and replicate it, this is in direct analogy with the way we can run the tape of life and every time a different rational agent will emerge as the "better-to-fit". The principles of recombination, selection, and mutation are basically "operators" in the algorithm to generate new individuals.

Turing versus non-Turing; abstraction versus implementation

This aspect is more speculative and reflects a general attitude towards computation in general. Why is evolutionary computation so special? Some theoretical results suggest that evolutionary Turing machines may are more expressive than Turing machines—at an abstract level. The so-called "Turing Evolutionary machine" is more expressible than an ordinary Turing machine, and its output can converge to the output of an universal Turing Machine. More importantly, the evolutionary Turing machine can solve the TM-unsolvable halting problem using non-algorithmic means (Eberbach, 2005). Generalizing computation to a non-Turing aradigm would

⁵I will follow here mainly Eberbach (2005). See also Pudlk (2001).

provide novel and unexpected epistemological results. Unlike Turing machines, the theory of Evolutionary Turing Machines is relatively unknown to the philosophical community. Eberbach has showed that evolutionary computation can be non-algorithmic, can evolve non-recursive functions and that an evolutionary Turing machine can solve the TM unsolvable halting problem of a UTM. "They are specific metaalgorithms (i.e., algorithms operating on other algorithms) with no restriction on their domain and some (rather historical) restriction on evolutionary algorithms that they have to be probabilistic, population-based, and using fitness function." Eberbach (2005). Practical implementations of evolutionary computation are approximations of Turing machines and they are heavily restricted to time and resources of concrete implementations.

Conclusion

With its "upward epistemology", evolutionary computation applied to discovery is a promising new tool for future scientific projects. Evolutionary computation and genetic algorithms in particular, anticipate the way scientific methodology and knowledge may look in a couple of decades. And the philosopher of science cannot wait for the foreseeable moment of the informational singularity when artificial intelligence will compete with humans. My humble philosophical prediction is that evolutionary computation, or some more "evolved" offspring of it, will be there at the "singularity" party - if there shall be any.

References

- Barberousse, A., Franceschelli, S., and Imbert, C. (2007). Cellular automata, modeling, and computation.
- Bod, R. (2006). Towards a general model of applying science. *International Studies in the Philosophy of Science*, 20(1):5–25.
- Boon, M. (2006). How science is applied in technology. *International Studies in the Philosophy of Science*, 20(1):27–47.
- Curd, M. (1980). The logic of discovery: an analysis of three approaches. In Nickles, T., editor, *Scientific discovery, logic, and rationality*, volume 56. Springer.
- Darden, L. (1998). Anomaly-Driven theory redesign: Computational philosophy of science experiment. In Bynum, T. and Moor, J., editors, *The Digital Phoenix*. Blackwell, Cambridge.
- Eberbach, E. (2005). Toward a theory of evolutionary computation. *BioSystems*, 82(1):1–19.
- Fogel, D. B., editor (1998). *Evolutionary Computation: The Fossil Record*. Wiley-IEEE Press, 1 edition.
- Galison, P. L. (1996). Computer simulations and the trading zone. In Galison, P. and Stump, D. J., editors, *The Disunity of science: boundaries, contexts, and power*. Stanford University Press, Stanford Calif.
- Hanson, N. R. (1958). Patterns of Discovery: An Inquiry Into the Conceptual Foundations of Science. Cambridge University Press.
- Holland, J. H. (1975). Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. University of Michigan Press, second edition bradford books, 1992 edition.

- Jong, K. A. d. D. (2006). Evolutionary Computation. MIT Press: A Bradford Book, Cambridge MA, 1st edition.
- Keller, E. (2003). Models, simulation, and 'Computer experiments'. In Radder, H., editor, *The Philosophy of Scientific Experimentation*, pages 198–215. University of Pittsburgh Press.
- Koestler, A. (1959). *The Sleepwalkers: A History of Man's Changing Vision of the Universe*. MacMillan, Los Angeles.
- Kuhn, T. S. (1962). *The Structure of Scientific Revolutions*. University Of Chicago Press, 3rd (1996) edition.
- Kuorikoski, J. and Pyhnen, S. (2013). Understanding non-modular functionality: Lessons from genetic algorithms. *Philosophy of Science*, 80(5):637–649.
- Lab, C. M. (2009). Eurega.
- Langley, P. (1979). Rediscovering physics with BACON.3.

 Proceedings of the Sixth International Joint Conference on Artificial Intelligence.
- Langley, P., Bradshaw, G. L., Simon, H. A., and Zytkow, J. (1987). Scientific discovery: computational explorations of the creative processes. MIT Press, Cambridge, Mass.
- Laudan, L. (1980). Why was the logic of discovery abandoned? In Nickles, T., editor, *Scientific discovery, logic, and rationality*, volume 56. Springer.
- Meheus, J. and Nickles, T., editors (2009). *Models of Discovery and Creativity*. Springer, 1st edition. edition.
- Mitchell, S. D. (1997). Pragmatic laws. *Philosophy of Science*, 64(4 Supplement):S468–S479. Journal Article.
- Morrison, M. (2006). Applying science and applied science: Whats the difference? *International Studies in the Philosophy of Science*, 20(1):81–91.
- Muntean, I. and Wright, C. D. (2007). Autonomous agency, AI, and allostasis a biomimetic perspective. *Pragmatics & Cognition*, 15(3):485–513. Journal Article.
- Nickles, T., editor (1980a). Scientific discovery, case studies, volume 60. D Reidel Pub Co.
- Nickles, T., editor (1980b). *Scientific discovery, logic, and rationality*, volume 56. Springer.
- Pennock, R. T. (2000). Can darwinian mechanisms make novel discoveries?: Learning from discoveries made by evolving neural networks. *Foundations of Science*, 5(2):225–238.
- Pennock, R. T. (2007). Models, simulations, instantiations, and evidence: the case of digital evolution. *Journal of Experimental & Theoretical Artificial Intelligence*, 19(1):29–42.
- Pudlk, P. (2001). Complexity theory and genetics: The computational power of crossing over. *Information and Computation*, 171(2):201–223.
- Reichenbach, H. (1949). Experience and Prediction: An Analysis of the Foundations and the Structure of Knowledge. Literary Licensing, LLC.
- Schmidt, M. and Lipson, H. (2009). Distilling Free-Form natural laws from experimental data. *Science*, 324(5923):81–85.
- Simon, H. A. (1973). Does scientific discovery have a logic? *Philosophy of Science*, 40(4):471–480. ArticleType: research-article / Full publication date: Dec., 1973 / Copyright 1973 Philosophy of Science Association.

- Simon, H. A., Langley, P. W., and Bradshaw, G. L. (1981). Scientific discovery as problem solving. *Synthese*, 47(1):1–27.
- Srensen, M. H. (2004). The genealogy of biomimetics: Half a centurys quest for dynamic IT. In Ijspeert, A. J., Murata, M., and Wakamiya, N., editors, *Biologically inspired approaches to advanced information technology*, volume 3141 of *Lecture notes in computer science*, page 496. Springer, Berlin; New York. Book, Section.
- Thagard, P. (1998). Computation and the philosophy of science. In Bynum, T. and Moor, J., editors, *The Digital Phoenix*. Blackwell, Cambridge.
- Waltz, D. and Buchanan, B. G. (2009). Automating science. *Science*, 324(5923):43 –44.
- Wartofsky, M. (1980). Scientific judgement: Creativity and discovery in scientific thought. In Nickles, T., editor, Scientific discovery, case studies, volume 60. D Reidel Pub Co.
- Winsberg, E. (2009). Computer simulation and the philosophy of science. *Philosophy Compass*, 4(5):835–845.
- Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding and selection in evolution. In *Proc of the* 6th International Congress of Genetics, volume 1, page 356366.

Steps Toward a Modular Library for Turning Any Evolutionary Domain into an Online Interactive Platform

Paul Szerlip and Kenneth O. Stanley

Dept. of EECS (Computer Science Division), University of Central Florida, Orlando, FL 32816 pszerlip@eecs.ucf.edu, kstanley@eecs.ucf.edu

Abstract

Natural evolution inspires the fields of evolutionary computation (EC) and artificial life (ALife). A prominent feature of natural evolution is that it effectively never ends. However, most EC and ALife experiments are only run for several days or weeks at a time. Once an experiment concludes, reproducing, observing, or extending the results often requires considerable effort. In contrast, some Collaborative Interactive Evolution (CIE) systems, e.g. Picbreeder, were designed to preserve results as potential stepping stones to build upon later while taking advantage of human insight to solve challenging problems. Traditionally, building long-running and open experiments similar to Picbreeder presents a complex and timeconsuming software challenge. To reduce this challenge and thereby remove the barrier to situating almost any experiment within an interactive online framework, this paper presents the initial prototype for Worldwide Infrastructure for Neuroevolution (WIN). Built in the model of Picbreeder, WIN is a modular library for significantly reducing the complexity of creating fully persistent, online, and interactive evolutionary platforms for any new or existing domain. WIN Online, the public interface for WIN, provides an online collection of domains built with WIN that lets novice and expert users browse and meaningfully contribute to ongoing experiments. Two example experiments in this paper demonstrate WIN's potential to quickly bootstrap any evolutionary domain with online and interactive capabilities.

Introduction

The fields of evolutionary computation (EC) and artificial life (ALife) are inspired by the products of natural evolution. As researchers in these areas, we hope to reproduce or one day exceed natural evolution's most inspiring qualities, for example through *open-ended evolution* (Channon, 2001; Maley, 1999) or evolving a diverse collection of virtual morphologies (Auerbach and Bongard, 2012; Hornby and Pollack, 2002; Lehman and Stanley, 2011; Sims, 1994; Szerlip and Stanley, 2013). However, almost all EC and ALife experiments fall short on one very salient feature of natural evolution: that it effectively never ends. Natural evolution perpetually branches from previous discoveries in a complex and never-ending process.

In stark contrast, presently, a research group that evolves e.g. a controller for a biped walking robot (Hein et al., 2007;

Reil and Husbands, 2002) typically will report the result, publish a version of the code, retire the experiment, and move on to new domains. Although the experimental results are indeed published, the ability to reproduce, observe, or extend those results generally requires considerable effort outside of the originating group. Yet in natural evolution the stepping stones that lead to the greatest discoveries almost never foreshadow the circuitous and dramatic discoveries that follow them. For instance, who could predict that a flatworm would one day evolve to become a human being (Raff, 1996)? Every experimental artifact left behind is a potential stepping stone lost forever.

Interestingly, there are unique experimental systems that attempt to collect such potential stepping stones. Collaborate Interactive Evolution (CIE) systems are designed to allow contributions from multiple users over the course of the experiment (Szumlanski et al., 2005). One such CIE system, Picbreeder, a genetic art program for breeding pictures, was explicitly designed to allow users to branch collaboratively from previously discovered results in the space of pictures (Secretan et al., 2011). That is, every picture evolved in Picbreeder is either a descendant of another picture inside the system, or started from a simple random starting point. Additionally, the system is still active after seven years, and every picture discovered is available online to continue extending at http://picbreeder.org/.

In this way, while Picbreeder can only evolve pictures, it provides precedent for evolutionary systems capable not only of preserving stepping stones accessible to the community, but also taking advantage of of human insight to search the space of all artifacts for the most meaningful. Yet the benefit of human intuition does not only apply to search spaces with hard to define or subjective optimization metrics. In fact, recent experiments suggest that humans are capable of interleaving their own intuition with automated algorithms, yielding better results than the automated algorithms can alone (Bongard and Hornby, 2013; Woolley and Stanley, 2014). In an ideal world, any experiment would have the ability to leverage human intelligence to aid the search while also allowing discovered results to

remain accessible to the community in perpetuity for future extension by humans, or by some clever algorithm.

Unfortunately, building experiments capable of interaction across the Internet is complex and time consuming. In fact, Picbreeder took over a year to build the online infrastructure (Secretan et al., 2011). Though web technologies have progressed in the years since Picbreeder was built, it would still take considerable effort to build such an infrastructure around any arbitrary experiment.

To begin to address this gap, this paper presents the initial prototype for Worldwide Infrastructure for Neuroevolution (WIN), a library that significantly reduces the complexity of creating fully persistent, online, and interactive (or automated) evolutionary platforms around any domain. The main outcome is to demonstrate that WIN can quickly ramp up CIE domains as well as effectively extend experimental domains that were not originally designed to be persistent or part of a CIE application. Yet the ambitions of WIN go beyond these example domains. At present, most researchers operate within isolated groups. The long term aim of WIN is to make it trivial to connect any individual or lab platform to the world, providing both a stream of online users, and archives of data and discoveries for later continuation. In short, the goal is to be able to continue where someone, or perhaps some algorithm, left off.

Background

In the fields of ALife and EC, many tools exist to assist researchers in constructing new experimental domains. Software packages like the Java-based ECJ suite (Luke, 2010) provide a collection of classes and data structures to solve common problems typically encountered when constructing new experiments (e.g. building a user interface and visualizing results). Other efforts in the community aim to enhance a communal pool of resources for collecting and sharing results. For example, the ALife Zoo aims to take advantage of cloud computing to host a shared platform for running ALife simulations (Hickinbotham et al., 2013). The Virtual Complexity Lab (VLab) is an online resource that aggregates a variety of ALife simulations to stimulate interest in the field (Green, 2007).

Together, these platforms represent a significant contribution to the community, helping to proliferate new domains and advanced simulations and to share improved results. However, there remains an open opportunity in ALife for additional platforms geared towards enhancing the contributions of laymen as well as academics.

The idea that casual human users can aid serious scientific endeavors has gained credibility in recent years with a number of online citizen science (Crowston and Prestopnik, 2013; Newman et al., 2012) projects in which users who often are not scientific experts are crowd-sourced to yield results that in some cases would be impossible to achieve in another way. Within evolutionary robotics, Bongard (2013)

created a simulator that sets precedent for harnessing amateur users for crowd-sourcing intricate robot controllers. Yet the benefits of crowd-sourcing can apply to more than any one domain.

There is thus a need for software that goes beyond organizing domains, simulations, or resources in the community. Such software could reconfigure scientific experiments to explicitly accumulate the results of the past in a *publicly* accessible format where they can be reproduced, observed, and extended with minimal effort.

WIN Architecture

To fully address the needs of the community, WIN serves as an active collaborative tool that not only stores past results, but presents them immediately for further exploration and extension. By design, WIN is conceived as both a platform and a service. The platform is a set of libraries and tools to assist in enabling any domain to allow access to its data online by algorithms or users, while the service aggregates a growing collection of ongoing experiments built with the WIN platform. To aid development, the platform is designed to significantly reduce the programming burden of making experimental data available, and provides methods for attaching any domain to the worldwide repository of experiments. On the other hand, the service is the public interface to all the collected domains, allowing interested users to freely browse available experiments linked by the WIN platform. Together, the WIN platform and service aim to amplify the collective effort of the community by reducing the work required to open any search space to both academics and laymen alike.

In more detail, the WIN platform is built in the model of Picbreeder but with an eye towards more general applications. Recall that one of WIN's goals is to integrate easily with *any domain*. As such, to prevent being too unwieldy to gracefully integrate existing domains, WIN's architecture is highly modular. Built on top of the JavaScript library Node.js (Dahl, 2009), WIN is a lightweight and expandable collection of Node.js packages that are optionally included for any domain. Keeping the core WIN library minimal but extensible, WIN avoids becoming a one-size-fits-all package. Overall, the WIN platform is a small set of libraries for handling storage, retrieval, and cataloging of complex chains of research artifacts similar to how genotypes are stored by the Picbreeder web service.

Modules

It is important to note that WIN is not confined only to organizing research data. To enable more functionality, a driving concept of the platform is the ability to create *WIN modules*, which are event-driven Node.js packages that plug in to the WIN framework. For example, the two domains demonstrated later in this paper extend WIN to include modules for

user interface elements and for managing automated evolutionary searches. A benefit of these modules is that they can be reused.

Borrowing from the event-driven design paradigm of Node.js, each WIN module specifies the events to which it responds as well as any events required from other modules. The overall WIN framework handles passing these generic messages and events between modules, and routing the responses to the appropriate places. While the exact technical details of how WIN handles message passing is available in the open-source repository https://github.com/OptimusLime/win-backbone, it is important to understand the implications of constructing the framework as a generic message passer.

There are two significant advantages to building WIN as a collection of event-driven modules. Primarily, any researcher in the community can bootstrap their own experimental work by extending any relevant WIN modules. Because each module only responds to a select set of events, augmenting a module does not require understanding the implementation details, but rather a higher level understanding of how to combine those events to add new functionality.

Second, the design enables the concept of module swapping. Due to the structure of Node.js, modules may request an event response without knowing *a priori* what module will respond. Researchers can take advantage of this feature by building modules that perform the same overall function powered by entirely different algorithms.

Saving Data in WIN

However, simply making WIN modular and event-driven is not a panacea for integrating WIN with any domain. Each evolutionary experiment can have distinct genetic encodings with custom algorithms to mutate and cross over genotypes, or special methods for exploring the search space.

To directly address these software challenges, the key insight behind WIN's architecture is to be agnostic to the processes generating the data, and instead focus on the form the experiment's data will take. In effect, WIN inverts the typical relationship between experiments and the data produced. As researchers, the primary focus often revolves around what algorithms and encodings produce the collected data. In contrast, WIN is primarily concerned with how the data is structured, which is defined a priori by the researcher for each experiment.

Formally, to maintain data with varying attributes and sizes, WIN enlists the JSON format (Crockford, 2006), as well as the JSON Schema specification for data validation (Galiegue and Zyp, 2013). JSON describes a data format for building complex data objects as the composition of a smaller set of universal data structures (e.g. strings, numbers, arrays and dictionaries). Similarly, the JSON Schema definition specifies a template language to describe the structure of JSON data for facilitating data validation. Essen-

tially, each JSON schema outlines a contract describing the format that data can take, which enables an application to validate that incoming data objects match the same format. Inside WIN, the JSON schema defined by the researcher dictates the internal structure of the data being stored. Before permanent storage, all data being saved by WIN is validated against the expected format. Currently, WIN employs the NoSQL document-database MongoDB for long term storage and retrieval, which is a natural fit for storing JSON data (Plugge et al., 2010).

Note that because WIN utilizes Node.js and JavaScript for its underlying functionality, it is more convenient to select JSON for data formatting over other storage types, e.g. XML. Regardless of the specific storage format, the key required property is the ability to represent a wide variety of data configurations, which enhances the ease of use when integrating new domains onto the platform. Importantly, WIN can store research artifacts while remaining encodingagnostic. Figure 1 shows how different genetic encodings can be represented as simple JSON schema, all of which can be saved by WIN. By default, a JSON schema is required for saving objects in WIN, but the schema specification is a simple and extensible template for saving any type of data. Interestingly, by design of the JSON format, data with almost any conceivable form can be stored by WIN, potentially opening the platform to support most genetic encodings actively researched by the community.

Phylogenies

Ultimately, when collecting research artifacts from evolutionary domains, the relationships among the data can tell an important story about how a domain solution was discovered. In evolutionary computation, experimental data commonly has a parent-child relationship, wherein one object is considered the direct descendant of another. By chaining a collection of objects and their relatives, an artificial phylogeny can be constructed. Previously, Woolley and Stanley (2011) investigated the Picbreeder phylogeny to understand why fitness-based automated evolution was having significant difficulty attempting to recreate images already evolved through interactive evolution by the users of Picbreeder.

Crucially, the Picbreeder phylogeny is likely not the only phylogeny capable of informing scientific research, but it is the only phylogeny available online. Of all the previous evolutionary experiments conducted, representing thousands of published papers, the lost phylogenies of those experiments may have contained potential treasure troves of information.

To account for this potential, WIN not only stores artifacts created by evolution, but simultaneously tracks the relationships among the data as well. Practically, tracking relationships in the data thereby requires a minimal amount of additional work by the researcher. By default, WIN attaches a unique identifier to each object saved internally. Therefore, to enable tracking connections in the data, each research

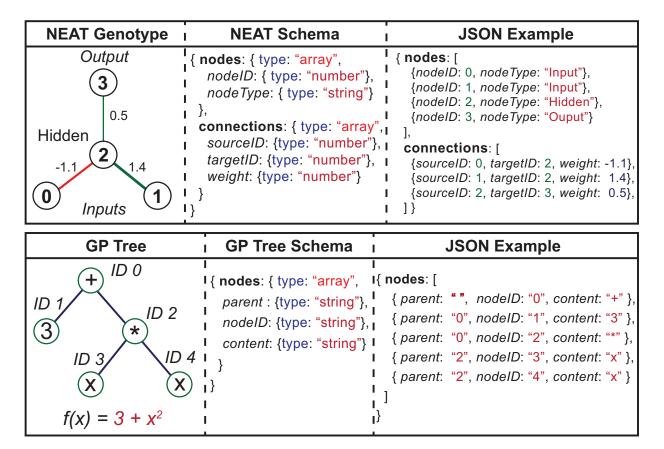


Figure 1: **Example Schema in WIN** Shown here are examples of two potential encodings and the corresponding JSON format for saving inside WIN. At top, a NEAT Genotype describes a *compositional pattern producing network* (CPPN) with four nodes and three connections (Stanley, 2007). Below, a GP-Tree (Koza, 1998) representing the function $f(x) = 3 + x^2$ is shown. For both figures, the middle column describes the expected composition of the data sent to WIN for the purpose of validation.

artifact being saved must provide an accompanying list of identifiers representing the object's parents. This additional parental list is enough to create a map of the ancestry across all artifacts.

As an experiment accumulates data, each object stored by WIN represents a potential branching point for future research with a traceable history of where the data originated. In this way, WIN aims to be a stepping stone accumulator for any evolutionary domain, allowing previously-finite experiments to exploit the possibilities of never-ending search.

WIN Online

While the WIN platform is designed to be minimally intrusive, WIN as a service, or WIN Online, aims for a larger role in the ALife and EC community. Accordingly, WIN Online is the public face to a worldwide repository of ongoing experiments for any evolutionary or artificial life domain integrated with the WIN platform. Recall that the data saved by WIN represents potential starting points for new interactive or automated searches. Thus WIN Online becomes a place where users who are interested in the ALife and EC com-

munities can participate in academic domains without prerequisite domain knowledge and immediately start a fresh evolutionary branch from existing results.

Incoming users are first presented with an active list of domains hosted by WIN Online. A small text description is included for each domain allowing the user to choose the most relevant to their interests. For domains that run entirely in the browser users of WIN Online may seamlessly transition from browsing all domains to browsing the current collection of artifacts contained within the selected domain. Depending on the programming language of the domain, the user may need to download a desktop client to access the experiment. The two examples demonstrated in the next section provide an example of what the WIN Online user experience is like for domains that operate both in and out of the browser.

For researchers, WIN Online acts as a potential resource for attracting users and crowd-sourcing new domain solutions. Note that researchers who prefer not to allow WIN Online to host their research data may host data on their own servers and supply an external link for WIN Online. To assist in creating an online collection of experiments, any domain built with WIN can integrate into the online repository with minimal additional effort. As discussed in *Saving Data in WIN*, internally, WIN utilizes the MongoDB database for storage. Taking advantage of a MongoDB feature, domains linked to WIN Online are given a separate database within the global MongoDB database hosted by WIN Online. Additionally, WIN Online handles configuration of the REST API required to access the newly created domain database and its contents.

Utilizing these features, WIN Online can provide a catalog of cutting edge ALife research with the potential to provide a steady stream of users to new research domains. A prototype for WIN Online that includes links to the two domains described next is accessible at http://winark.org/.

Example Domains

The key to enticing a community to develop for a platform like WIN is to show that new domains can be added easily and systematically. By demonstrating this point through two domains that would otherwise be highly challenging to put online without WIN, this section (and its corresponding demonstrations online) not only shows the posssibilities that WIN creates but also provides a reference for future developers aiming to extend WIN with more domains.

The first is an HTML/JavaScript clone of Picbreeder (Secretan et al., 2011). The second is IESoR, a Sodaraceinspired (McOwan and Burton, 2005) domain for evolving two-dimensional morphologies capable of ambulation (Szerlip and Stanley, 2013). Some resulting phenotypes from both Picbreeder and IESoR are shown in figure 2. To avoid confusion, the version of Picbreeder integrated with WIN will be called win-Picbreeder, and the IESoR variant will be called win-IESoR. Both examples are currently accessible through WIN Online at http://winark.org/. Together, the Picbreeder and IESoR domains aim to cover a broad range of interests inside the ALife community. Picbreeder is a proxy for domains that are more suited towards Interactive Evolutionary Computation (IEC) (Takagi, 2001) and difficult to optimize. In contrast, IESoR is a control-based domain that runs multiobjective search to discover new ambulatory creatures, i.e. it is designed for automated optimization.

Notably, IESoR was originally coded and simulated in JavaScript. For this demonstration, the IESoR simulation was rewritten in C++, and wrapped by a JavaScript WIN module. Writing the IESoR simulation in a non-scripting language demonstrates that even though the WIN system is written in JavaScript, WIN modules can be written in a native programming language like C++. Supporting native and scripting languages allows WIN modules to enjoy the performance benefits of native code when required while taking advantage of the ease of scripting when optimal performance

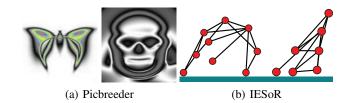


Figure 2: **Products of Evolution.** Picbreeder artifacts (a) are $n \times n$ pixel images where the RGB values are constructed from the outputs of a CPPN (Stanley, 2007). (b) A pair of two-dimensional ambulating creatures are shown from the IESoR domain (Szerlip and Stanley, 2013).

is not necessary.

To accommodate the two example domains, several new WIN modules were built to augment the features described in the WIN Architecture section. Importantly, both Picbreeder and IESoR depend on two new modules, winhome and win-gen. Win-home is a user interface (UI) module that mimics the web portion of Picbreeder, which is intended to provide a simple user interface for domains hosted on WIN. The module includes a homepage template similar to Picbreeder for displaying new or interesting artifacts published by users. The win-gen module aids in creating new objects matching a predefined data format and in particular for generating the underlying indirect encoding (CPPNs) powering both domains (Stanley, 2007). An exhaustive list of modules created so far is available online at http:// winark.org/modules, and the exact programming details of these modules are all accessible and open-source. Developers who are new to WIN will have the benefit of the already-included domains and modules built for both demonstrations that thereby serve as templates for new experiments in WIN.

Picbreeder

Originally, the Picbreeder website was written in PHP, while the Picbreeder evolution client responsible for generating the images was written in Java. For a more modern approach, win-Picbreeder is written in HTML5 and JavaScript, which has the added benefit of running in any browser without plugins. Because WIN in conceived in the mold of Picbreeder, it follows that win-Picbreeder is among the easiest projects to integrate with WIN.

To match the major features of the original Picbreeder, win-Picbreeder needs a homepage, an IEC user interface, and the ability to store, retrieve, and generate image artifacts. As described in *Saving Data in WIN*, WIN offers an encoding-agnostic method to store, retrieve, and generate custom schema. In the case of Picbreeder, the schema simply contains a NEAT genotype, i.e. the CPPN, conforming to the structure depicted in figure 1a, along with user tags describing the final phenotype image in a few keywords.

Therefore, the main issue for creating win-Picbreeder is how to connect the IEC user interface to the existing WIN framework for saving artifacts.

In WIN, the solution is fairly simple. Recall that WIN's main architecture is event-driven. As the user explores the domain through the IEC interface, whenever an artifact must be displayed, an event is passed to WIN to generate a new artifact from the currently selected parents. WIN routes this message to the win-gen module described above that is responsible for creating new artifacts, and a new win-NEAT module creates NEAT genotypes and the corresponding CPPNs by combining the provided parent genotypes. After the IEC interface receives the new artifact object(s), domain-specific Picbreeder code converts the NEAT genotype in the artifact to the displayed picture in the interface through WebGL.

The majority of complexity in Picbreeder is in the storage and retrieval of evolutionary data. Because WIN is designed specifically to handle the data management task, the rest of the win-Picbreeder application is lightweight compared to the original Picbreeder's code. Altogether, win-Picbreeder effectively demonstrates a simple WIN application; the code is available online at https://github.com/OptimusLime/win-Picbreeder.

The exciting point about adding this domain is that numerous IEC-based domains can easily be constructed and put online simply by deriving them from the win-Picbreeder code. That is, with minimal effort researchers can create services like Genetic-Programming-Picbreeder, L-Systems-Picbreeder, or any such conceivable variant.

IESoR

In the original IESoR, an automated NSGA-II multiobjective search (Deb et al., 2002) evolved functional two-dimensional ambulating morphologies similar to the creatures depicted in figure 2b. In the win-IESoR application, the same multiobjective search algorithm operates with one important difference: the human user is included in the loop. Instead of running an automated algorithm for a fixed period of time and collecting the results, win-IESoR interleaves occasional choices by the user with shortened multiobjective searches and returns the most promising individuals from which the user can further evolve. Woolley and Stanley (2014) demonstrated recently the ability of human choices interleaved with search to outperform even automated algorithms. WIN is uniquely positioned to make this kind of interleaved search easy to integrate into any domain.

Because interactive evolution was not part of the original IESoR domain, it was not necessary to build win-IESoR with user interaction. However, explicitly including user interaction with win-IESoR helps to highlight a deeper purpose behind the infrastructure of WIN. If humans have the ability to add insight and utility to search and WIN makes the process of including a human in the loop relatively pain-

less, then the hope is that researchers in the future will not need to hesitate to take advantage of human insight whenever it is appropriate.

To enable user interaction with an automated search, a new WIN module named win-NSGA was created to handle the complexity of this interleaving search. Following precedent in Woolley and Stanley (2014), after a certain number of viable candidates are found through automated evolution, the search returns the results to the user interface for human selection. The main takeaway is that win-IESoR demonstrates that the WIN platform is capable of executing a search process that involves both an automated algorithm and a human.

In win-IESoR, the data saved by WIN includes additional components beyond those saved by win-Picbreeder. The data contains both a NEAT genotype to define the creature morphology as well as the parameters and objects inside of the two-dimensional physics engine (e.g. the ground, gravity, and friction). Along with the win-home user interface templates, a new UI module for interleaving search was created inspired by the user interface elements from Woolley and Stanley (2014).

Altogether, the final program is a collection of HTML5 user interfaces connected through Node.js to the IESoR simulations being run in C++. While the interface for interactive evolution in win-IESoR is still under construction, the results of an initial set of evolutionary experiments within win-IESoR are browsable through the WIN Online service.

A major benefit of WIN is that there is no need to write and re-write the same code when someone in the community has already written it. Thus all the infrastructure written for win-IESoR that allows integrating multiobjective searches with interleaved human selection can now be applied to any other ALife domain, which should help to accelerate research in such systems significantly.

WIN Phylogenies

An important element of win-Picbreeder and win-IESoR is the ability for the user to decide when an artifact should be saved for the public record, i.e. published. To save discovered artifacts, the user clicks a publish button on the user interface that dispatches an event to WIN to store the chosen object in the WIN database. On the path to the published artifact, the user has likely made multiple selection choices or run several evolutionary searches before choosing to publish. WIN allows the researcher to configure how much from these intermediate choices and associated meta-information is stored in WIN.

Recall that regardless of configuration, WIN always tracks parent-child relationships among the published artifacts. As users interact with a domain, each published object adds a single branch to the tree of artifacts inside the database. At any point during the ongoing experiment, researchers may compile part or all of the published objects

into an artificial phylogeny. WIN assists in constructing phylogenies by providing methods for retrieving the full parent and children objects for any artifact within the database. Researchers can repeatedly query these methods to unravel a chain of research artifacts and construct a phylogeny.

To highlight this WIN feature, phylogenies for win-Picbreeder and win-IESoR are shown in figure 3a and 3b, respectively. For both applications, the data represent a cross-section of artifacts generated by a single researcher. It is important to note that these artificial phylogenies do not represent static aggregated results, but rather collections of potential stepping stones that are all available to branch from currently in the ongoing experiments. Both win-Picbreeder and win-IESoR are open for browsing artifacts through WIN Online at http://winark.org/. Win-Picbreeder already supports contributing new artifacts, while win-IESoR is a prototype that allows browsing artifacts generated during initial experiments. A complete win-IESoR interface for new public contributions is under construction.

In contrast to traditional experimental results, the results in this section do not embody the conclusion of an experiment. Instead, these two domains help to exemplify the promise of WIN to aid researchers in creating experiments that may potentially never end. The resulting phylogenies are an important step towards validating that the WIN platform can deliver on this promise.

Discussion and Future Work

WIN is currently a prototype that will be brought to eventual maturity through more documentation, complete illustrated examples, and a sophisticated WIN module manager. As the WIN library matures, the documentation will be updated to reflect the nature of the library. Similarly, an increasing collection of coding examples will provide a consistently improving reference to developers working with the platform.

Building a tool that can organize and unlock the data within any arbitrary evolutionary domain is ambitious, but the WIN prototype hints at how it could be possible. The lightweight design of WIN aspires to create a new type of research collection full of open domains assembled together for easy and perpetual access through WIN Online. Moreover, WIN opens the door to new and expansive research questions. WIN empowers us to start imagining what it would be like one day to have a living archive of research where every object from every evolutionary experiment in progress is actively preserved with the full context of its ancestry known and its future open to exploration. What phenomena might emerge from the data when evolution is no longer run by labs in isolation for days or weeks, but by many labs in parallel across the world for years?

Conclusion

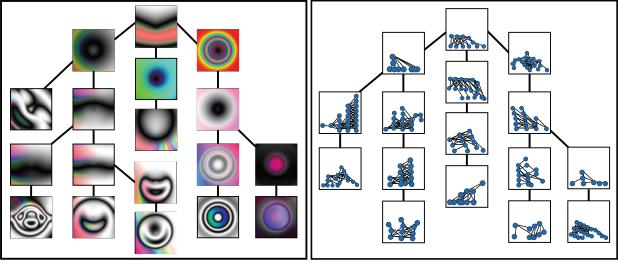
This paper represents a milestone on the path towards building an infrastructure to give researchers from evolutionary computation and artificial life the ability to quickly make their research available to users online, who can then genuinely contribute. Similarly, WIN Online aggregates domains built with the WIN platform into a single source for new and exciting research that allows users to browse and participate. The WIN library demonstrates that it is indeed possible to construct an online and interactive platform around almost any evolutionary domain, which may help to proliferate the availability of experiments that continually collect potential stepping stones for extension and thereby effectively never end.

Acknowledgments

This work was supported by the National Science Foundation under grant no. IIS-1002507. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of the National Science Foundation.

References

- Auerbach, J. E. and Bongard, J. C. (2012). On the relationship between environmental and morphological complexity in evolved robots. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2012)*, pages 521–528, New York, NY. ACM Press.
- Bongard, J. C. (2013). Ludobots website. The Ludobots software is publicly available at http://www.uvm.edu/ ludobots/.
- Bongard, J. C. and Hornby, G. S. (2013). Combining fitness-based search and user modeling in evolutionary robotics. In *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, GECCO '13, pages 159–166, New York, NY, USA. ACM.
- Channon, A. (2001). Evolutionary Emergence: The Struggle for Existence in Artificial Biota. PhD thesis, University of Southampton.
- Crockford, D. (2006). RFC 4627 The application/json Media Type for JavaScript Object Notation (JSON). Technical report.
- Crowston, K. and Prestopnik, N. R. (2013). Motivation and data quality in a citizen science game: A design science evaluation. In 2013 46th Hawaii International Conference on System Sciences (HICSS), pages 450–459. IEEE Press.
- Dahl, R. L. (2009). Node.js software package. The Node.js software package is publicly available at http://nodejs.org/.
- Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.
- Galiegue, F. and Zyp, K. (2013). Json schema: core definitions and terminology draft-zyp-json-schema-04. Working Draft.
- Green, D. (2007). Vlab website. The VLAB website is is publicly available at http://vlab.infotech.monash.edu.au/.
- Hein, D., Hild, M., and Berger, R. (2007). Evolution of biped walking using neural oscillators and physical simulation. In RoboCup 2007: Proceedings of the International Symposium, LNAI. Springer.
- Hickinbotham, S., Weeks, M., and Austin, J. (2013). The alife zoo: cross-browser, platform-agnostic hosting of artificial life simulations. In *Proceedings of the European Conference on Artificial Life (ECAL-2013)*.



(a) win-Picbreeder (b) win-IESoR

Figure 3: **WIN Phylogenies.** The tree of artifacts in (a) and (b) represent artificial phylogenies resulting from the efforts of a single researcher. In (a), each square represents a published image inside win-Picbreeder, while each connection represents a direct relationship between the images. Each image in (b) illustrates the starting morphology of a two-dimensional ambulating creature evolved by win-IESoR. Though images are linked by a single connection, there may have been multiple human selections and potentially hundreds or thousands of automated evaluations in each branch of the tree. Both phylogenies contain artifacts that are currently available for browsing in win-Picbreeder and win-IESoR by visiting WIN Online at http://winark.org/.

- Hornby, G. S. and Pollack, J. B. (2002). Creating high-level components with a generative representation for body-brain evolution. *Artificial Life*, 8(3).
- Koza, J. R. (1998). Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press, Cambridge, MA.
- Lehman, J. and Stanley, K. O. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223.
- Luke, S. (2010). The ECJ Owner's Manual A User Manual for the ECJ Evolutionary Computation Library, zeroth edition, online version 0.2 edition.
- Maley, C. C. (1999). Four steps toward open-ended evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, pages 1336–1343.
- McOwan, P. W. and Burton, E. J. (2005). Sodarace: Adventures in artificial life. In *Artificial Life Models in Software*, pages 97–111. Springer.
- Newman, G., Wiggins, A., Crall, A., Graham, E., Newman, S., and Crowston, K. (2012). The future of citizen science: emerging technologies and shifting paradigms. *Frontiers in Ecology and the Environment*, 10(6):298–304.
- Plugge, E., Hawkins, T., and Membrey, P. (2010). *The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing.* Apress, Berkely, CA, USA, 1st edition.
- Raff, R. A. (1996). *The Shape of Life: Genes, Development, and the Evolution of Animal Form.* The University of Chicago Press, Chicago.
- Reil, T. and Husbands, P. (2002). Evolution of central pattern generators for bipedal walking in a real-time physics environment. *IEEE Transactions on Evolutionary Computation*, 6(2):159–168.

- Secretan, J., Beato, N., D.Ambrosio, D. B., Rodriguez, A., Campbell, A., Folsom-Kovarik, J. T., and Stanley, K. O. (2011). Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary Computation*, 19(3):345–371.
- Sims, K. (1994). Evolving 3D morphology and behavior by competition. pages 28–39. MIT Press, Cambridge, MA.
- Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines Special Issue on Developmental Systems*, 8(2):131–162.
- Szerlip, P. and Stanley, K. O. (2013). Indirectly encoded sodarace for artificial life. In *Proceedings of the European Conference on Artificial Life (ECAL-2013)*.
- Szumlanski, S. R., Wu, A. S., and Hughes, C. E. (2005). Collaborative interactive evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) Poster Session*, New York, NY. ACM Press.
- Takagi, H. (2001). Interactive evolutionary computation: Fusion of the capacities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296.
- Woolley, B. G. and Stanley, K. O. (2011). On the deleterious effects of a priori objectives on evolution and representation. In GECCO '11: Proceedings of the 13th annual conference on Genetic and evolutionary computation, pages 957–964, Dublin, Ireland. ACM.
- Woolley, B. G. and Stanley, K. O. (2014). Novel human-computer collaboration: Combining novelty search with interactive evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2014)*, New York, NY, USA. ACM. To appear.

Effects of Personality Distribution on Collective Behavior

Brent E. Eskridge¹ and Ingo Schlupp²

¹Southern Nazarene University, Bethany, OK 73008 ²University of Oklahoma, Norman, OK 73019 beskridge@snu.edu

Abstract

Optimizing group success is challenging for multi-robot systems, especially for large systems such as robot swarms where even simple individual interaction rules can lead to complex group behavior. Studies of natural systems have shown that heterogeneous groups can outperform homogeneous groups, especially when individual differences lead to role or niche specialization. This happens even when the individuals are seemingly identical. Although individuals within a group may appear physically identical, they can vary widely in their personality, which significantly affects their behavior. However, determining the most effective composition of personalities in a group is particularly difficult in natural systems given the ambiguities of animal personalities and the physical challenges of repeated evaluations. Using a biologicallybased collective movement model, we evaluate different personality distributions to determine their effect on the overall success of the group. Results show that although there are distributions that are clearly more effective than others, in many cases, there is a broad range of distributions that results in high group success. Furthermore, experiments using variable, or adaptive, personalities demonstrate that successful distributions are stable equilibriums as initially extreme distributions converge to these successful distributions as personalities change.

Introduction

In many robot swarms, individuals within the swarm are assumed to be heterogeneous, especially, and perhaps most importantly, with respect to their decision-making. In fact, heterogeneity is argued to be a requirement for a multi-robot system even to be considered a swarm (Şahin, 2005). Variations in either morphology or decision-making can present significant challenges given the complexities that arise as robots interact. However, such differences can also provide significant benefits (Couzin et al., 2005).

While most work involving these benefits focuses on morphological differences, of particular interest here are differences in the decision-making process, especially those that are adaptive. The ability for individuals in a group to adapt their decision-making to improve group-level performance is key to the usefulness of multi-robot systems in dynamic, real-world tasks. Recent research in animal groups has

shown that such systems exist in nature. Animals are known to exhibit sets of correlated traits referred to as personalities, that in part, determine their behavior (Sih et al., 2004). Furthermore, these personalities are known to change in response to both individual experiences and the distribution of personalities in the overall group (Dall et al., 2004; Montiglio et al., 2013). As such, animal personalities can provide inspiration as a means of integrating adaptive individual differences within a group to improve performance. Furthermore, we have previously shown that using a simple mechanism to introduce personality into a collective movement model can result in a significant increase in success (Eskridge et al., 2013). However, it is not well understood how the distribution of personalities affects group performance, nor how stable effective personality distributions are.

In the work presented here, we use a biologically-based collective movement model to explore how distribution of personalities, especially as they pertain to leaders and followers, affects the overall success of the group in initiating a movement. Simulations demonstrate that, for many of the groups evaluated, there is a wide range of personality distributions that results in high group success. Furthermore, the results demonstrate that these successful distributions are stable equilibriums. This is indicated not only by the wide range of successful distributions, but also by additional experiments using variable personalities in which extreme personality distributions converge to the more successful distributions as personalities change.

Related Work

Animal personalities are the focus of much recent research effort (Sih et al., 2004; Wolf et al., 2007). Personalities refer to correlated behavioral traits in animals that are frequently found together such as an insensitivity to social information and an increased likelihood of leaving the safety of cover to forage for food. The most commonly studied personality traits lie along the bold-shy continuum and have been observed in a variety of animal species (Frost et al., 2007; Harcourt et al., 2009b). Of particular interest to the work presented here is the fact that individuals with bold person-

alities are frequently leaders of collective behavior, while shy individuals are frequently followers (Dyer et al., 2009; Harcourt et al., 2009a; Kurvers et al., 2009, 2011). Personalities are known to be shaped by experience through mechanisms such as winner and loser effects (Hsu et al., 2006). In one case, bold losers became shyer and shy winners became bolder (Frost et al., 2007).

Personality differences between individuals can lead to differentiation in roles within a group. The combination of different personalities within a group and the associated roles assumed by different members have been found to improve the overall success of the group (Couzin et al., 2005; Dyer et al., 2009; Modlmeier and Foitzik, 2011; Modlmeier et al., 2012). Studies have shown that these personality differences can be stable and maintained over time (Dall et al., 2004; Oosten et al., 2010). One explanation for the maintenance of personality differences is through niche role specialization (Montiglio et al., 2013), with some theorizing that specialization occurs at the genetic level (Montiglio et al., 2013; Nonacs and Kapheim, 2007).

Although these personality differences are known to confer benefits to group performance, the effects of the distribution of these personalities within a group on that performance are not well understood. This is in part due to the complexity of the interactions between individual personalities, including the effects on group behavior through individual personalities (Magnhagen and Staffan, 2005; Cote et al., 2011) and the effects on individual behaviors by group personality (Cote et al., 2011). Furthermore, it is difficult to precisely measure the distribution of individual personalities in a group. In fact, few studies even report the specific distribution of personalities. Distributions that are reported consist of groups that are either randomly created in the lab (Magnhagen and Staffan, 2005), are drawn from environments that have unique fitness effects on personalities (Magnhagen, 2006), or do not classify personalities as bold and shy (David et al., 2011). While some studies have constructed groups with artificial personality distributions to evaluate the effects of different compositions (Sih and Watters, 2005; Pruitt et al., 2013), most did not systematically evaluate a range of different personality distributions to directly test the effects on overall group performance. However, Pruitt and Keiser (2014) recently observed that spider colony success in collective foraging was tied to the degree to which a leader is bold. Their results indicate that even when the number of bold and shy individuals were kept constant, groups were more successful when the difference between bold and shy personalities was large.

Methods

The simulations used for this work were performed using a modified version of a collective movement model developed through observations of collective movement attempts in a group of white-faced capuchin monkeys (Gautrais, 2010;

Petit et al., 2009), and was later confirmed in observations of sheep (Pillot et al., 2011). The model could be classified as a probabilistic finite state machine (PFSM) (Brambilla et al., 2013) and uses anonymous quorum-sensing to determine the rates of collective movement decisions. The anonymous quorum-sensing nature of the model means that leaders, whether successful or unsuccessful, cannot be recognized as such, and decision-making must rely on the number of participating individuals only. Despite being a collective movement model, actual movement through an environment is not a part of the model. Rather, the focus of the model is on the decision-making process that precedes a movement. Examples of such situations are found in nature where individuals exhibit notifying behaviors indicating a preferred direction of movement during a predeparture period (Pyritz et al., 2011).

Collective Movement Model

The collective movement model uses three rules to govern the decision-making process involved in starting collective movements (Petit et al., 2009; Gautrais, 2010). The first rule assumes that all individuals within the group can initiate a collective movement attempt with a constant rate of $1/\tau_o$, with $\tau_o=1290$. While this assumption may not hold for groups with dominant leaders, studies have shown that it is a viable assumption for egalitarian animal groups, such as the capuchin monkeys used in the model's development and robot swarms (Brambilla et al., 2013).

The second rule describes the rate at which followers join the collective movement attempt and is calculated by $1/\tau_r$. The time constant τ_r for the following rate is calculated using the following:

$$\tau_r = \alpha_f + \beta_f \frac{N - r}{r} \tag{1}$$

where $\alpha_f=162.3$ and $\beta_f=75.4$ are constants determined through direct observation, N is the number of individuals in the group, and r is the number of individuals following the initiator. As the number of individuals following the initiator increases, the rate at which individuals decide to join the movement also increases.

Not all initiation attempts are successful as initiators often cancel and return to the group. The third rule calculates this cancellation rate using the following:

$$C_r = \frac{\alpha_c}{1 + (r/\gamma_c)^{\varepsilon_c}} \tag{2}$$

where $\alpha_c=0.009$, $\gamma_c=2.0$, and $\varepsilon_c=2.3$ are constants determined through direct observation, and r is the number of individuals following the initiator. Simulations of the model include the implicit assumption that a successful collective movement requires all of the members of the group to participate since there is a non-zero probability of canceling even

if all but one member participates. While this is not necessarily the case in nature, cohesive collective movements are the primary objective of this work and, as such, incomplete movements are considered failures.

Integrating Personality

As with our previous work (Eskridge et al., 2013), personality was incorporated into the collective model using an individual-specific value referred to as a "k factor" (Gautrais, 2010). In the original model, this value was a constant and was used to investigate the effects of altering the rate at which individuals initiate, follow, and cancel. With the addition of the k factor, initiation attempts are calculated at the constant rate of k/τ_o , and the following and canceling rate calculations are calculated as follows:

$$\tau_r = \frac{1}{k} \left(\alpha_f + \beta_f \frac{N - r}{r} \right) \tag{3}$$

$$C_r = k \left(\frac{\alpha_c}{1 + (r/\gamma_c)^{\varepsilon_c}} \right) \tag{4}$$

where the variables are defined as before. Since this k factor can either increase or decrease the three decision-making rates, it was an ideal means of incorporating the effects of personality into the model.

Three important points were considered in integrating personality with the collective movement model. First, personality has been observed in natural systems to affect the events used in this model in different ways. For example, a bold personality should result in a higher initiation rate and lower following and canceling rates, while a shy personality should result in a lower initiation rate and higher following and canceling rates (Harcourt et al., 2009a). Second, the magnitude with which a shy personality affects the model should be the same as a bold personality so as not to bias effects of one personality value over another. Since k had a non-inclusive lower limit of zero, the non-inclusive upper limit of two was chosen to ensure balance. In the simulations described below, personalities used to calculate k were limited to the range [0.1, 0.9] to ensure these limits were satisfied. Lastly, although neither the original model nor the observations on which the model was based, discuss personality of the individual animals involved one can assume that the individuals could be classified as having either bold or shy personalities. Therefore, the integration of personality incorporated the concept of a moderate personality (p = 0.5)which produced the same equations as the original model. Larger personality values were interpreted as bolder personalities (e.g., p = 0.9) and smaller values were interpreted as shyer personalities (e.g., p = 0.1). To accentuate the effects of even minor differences in personality in values close to a moderate personality and minimize the effects of differences in extreme personalities, a personality p was converted to a corresponding k value using the following sigmoid function:

$$k = 2\left(1 + e^{\frac{0.5 - p'}{10}}\right)^{-1} \tag{5}$$

where p' is p for initiating and is 1-p for canceling and following. Although different parameter values could be chosen for each decision, the choice of a single parameter was made to avoid over-complicating the model.

Winner and loser effects are feedback mechanisms in which experience affects future success. They sometimes exist as a self-assessment mechanism that affects an individual's perception of its own abilities to the point where it can alter behavior without any actual change in ability (Fawcett and Johnstone, 2010). This self-assessment was incorporated into the model by updating the initiator's personality after every initiation attempt using the following standard update (or learning) rule (Bernstein et al., 1988; Katsnelson et al., 2012; Sutton and Barto, 1998):

$$p_{t+1} = p_t(1-\lambda) + \lambda r \tag{6}$$

where p_t was the initiator's personality before the movement, p_{t+1} was the personality after the movement, λ was the rate at which updates changed the personality, and r was the reinforcement value used to update the personality. When λ was low, the personality was primarily determined through long-term historical success and changes were minor. When λ was high, the personality was primarily determined through short-term success, namely the last initiation attempt, and changes from one attempt to the next were significant. For the simulations described here, a low value of lambda was chosen ($\lambda=0.02$) to emphasize long-term initiation success. For successful initiations, the reinforcement was r=1, while it was r=0 for unsuccessful initiations.

Numerical Implementation

Numerical simulations of the collective movement model were implemented in Java using a customized version¹ of the original algorithm (Gautrais, 2010). The time of each event was calculated as a random number drawn from an exponential distribution using the appropriate rate equation. As such, the simulations used continuous time events and not discrete time.

The original model was only evaluated with a group size of 10, but other work has shown that the success of collective movement initiations increases as the group size is increased, with most success differences present in group sizes of 50 or less and diminishing effects beyond a group size of 100 (Eskridge, 2012). To evaluate the effects of personality distributions as the number of individuals is scaled up, experiments using group sizes of 10, 20, 30, 40, 50, and

¹Simulation source code and data analysis scripts are available for download from https://github.com/snucsne/bio-inspired-leadership.

Group Size	Bold Individuals	Bold Percentage	Success Percentage
10	2	20%	86.9%
20	7	35%	91.7%
30	12	40%	93.6%
40	18	45%	94.4%
50	21	42%	94.8%
100	44	44%	95.5%

Table 1: Results of simulations with fixed personalities used to identify the most effective personality distributions of bold and shy personalities are shown.

100 were performed. To account for the fact that the model is probabilistic, fifty evaluations were performed for each group size, each with a different random seed. A single evaluation consisted of $2,000 \times N$ simulations, where N was the group size. Each simulation constituted a single attempt at a collective movement and ended in either success (all individuals participating in the movement) or the initiator canceling. Individual personality values were reset at the beginning of each evaluation and persisted from one simulation to the next. The model parameters noted above were the same as those used in the original model (Gautrais, 2010; Petit et al., 2009).

Results & Analysis

Two types of experiments were performed to evaluate the effects of the distribution of personalities on the overall group's success in initiating collective movements: *a*) fixed personalities with a fixed bold/shy distribution, and *b*) variable personalities with a fixed initial bold/shy distribution. The first type of experiment allowed for the identification of effects on the initiation success by different personality distributions. The second type allowed for evaluations on the stability of effective distributions and the ability of groups with suboptimal distributions to adapt.

Fixed Distribution

To find the most effective personality distribution of bold and shy personalities, referred to hereafter as the "optimal" distribution, a search of different distributions using fixed personalities was made. Figure 1 shows the percentage of bold personalities in the group versus the mean success for leaders initiating collective movements. Although the differences between many bold percentages, especially for larger group sizes, were not statistically significant, the bold percentage with the highest mean success was chosen as the "optimal" distribution. Note that evaluations using either all bold or all shy personalities performed the same. In fact, further analysis indicated that the results were identical to one another and results of previous simulations that did not use the personality modification.

Table 1 shows the details for evaluations with the highest mean success for group sizes of N=10, 20, 30, 40, 50, and 100. Note that while the "optimal" number of bold individuals in the group increased as the group size increased, the percentages fluctuated, especially for groups sizes N=40, 50, and 100. Since, as mentioned above, the differences between many distributions was not statistically significant, we cannot claim that these fluctuations are generalizable, but it does merit note.

Variable Distribution

Once the most effective fixed personality distributions of bold and shy personalities were identified, evaluations using variable personalities were performed. Figure 2 shows the percentage of bold personalities in a group at the beginning of the evaluation versus the end of the evaluation for each group size and a range of initial percentages of bold personalities. Other than a group size of N=10, groups were able to support a wide range of distributions of bold and shy personalities, including many that were much larger than the "optimal" distribution previously identified.

Figure 3 shows the initial percentage of bold personalities in the group versus the mean success for leaders initiating collective movements. For group sizes of N=10, 20, and 30, there was close agreement of the maximum success percentage between evaluations with variable personality and the most effective fixed personality distribution, denoted with a horizontal dashed line. However, simulations with variable personality had statistically significantly lower fitness than simulations with fixed personalities in the optimal distribution (Kolmogorov-Smirnov, p << 0.001). Interestingly, for larger group sizes, initial personality distributions with a greater percentage of bold individuals than the "optimal" (denoted with a vertical dashed line) performed consistently better than distributions closer to the most effective distribution found in using fixed personalities.

Figure 4 depicts representative personality histories for a group size of N = 40 and three different initial personality distributions. In Figure 4a, only 10% of the group was initially bold. However, within relatively few simulations, additional bold leaders emerged from within the group. Furthermore, significant variations in shy personalities were observed as individuals experienced temporary success in initiating movements, but ultimately failed to consistently succeed. In Figure 4b, 50% of the group was initially bold. Since this value was close to the "optimal" distribution (see Figure 3d), no transitions in personality from bold to shy or shy to bold were observed. Lastly, in Figure 4c, 90% of the group was initially bold. As the experiments with fixed personalities showed that groups with such extreme personality distributions are clearly suboptimal, failed bold initiators quickly adapted their personalities, and their roles, to be shy followers. Once this adaptation occurred, personality values stabilized for the remainder of the evaluation.

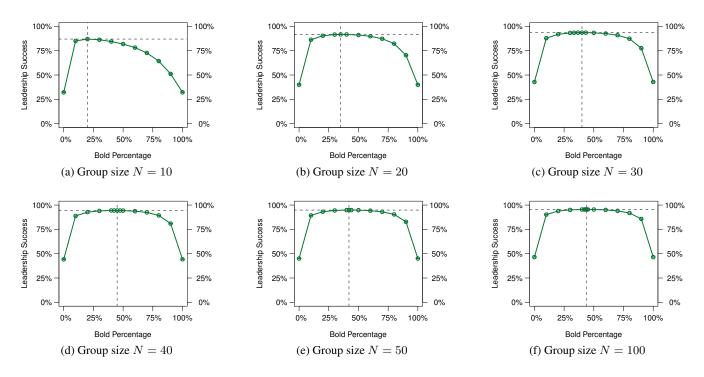


Figure 1: Plots of the percentage of bold individuals in the group versus the mean percentage of successful initiations for different group sizes in simulation treatments with *fixed* personalities and *fixed* distributions are shown. Horizontal and vertical lines denote the "optimal" percentage of bold individuals and the resulting mean success, respectively.

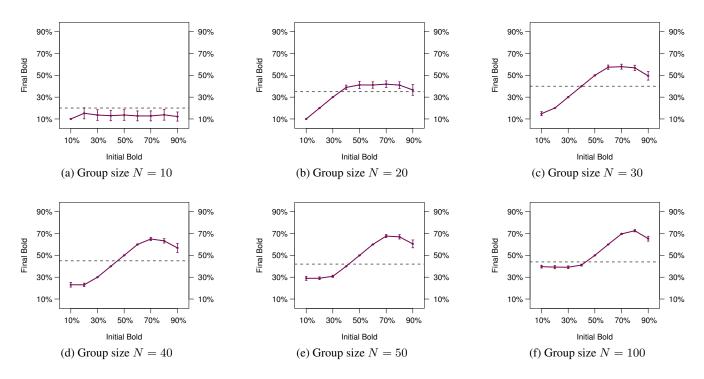


Figure 2: The mean percentage of bold personalities at the end of $2000 \times N$ simulations with *variable* personalities and fixed *initial* distributions are shown (mean/SD). Error bars representing the standard deviation are also plotted, but are often too small to see. Horizontal lines denote the most effective bold personality percentage found during fixed personality evaluations.

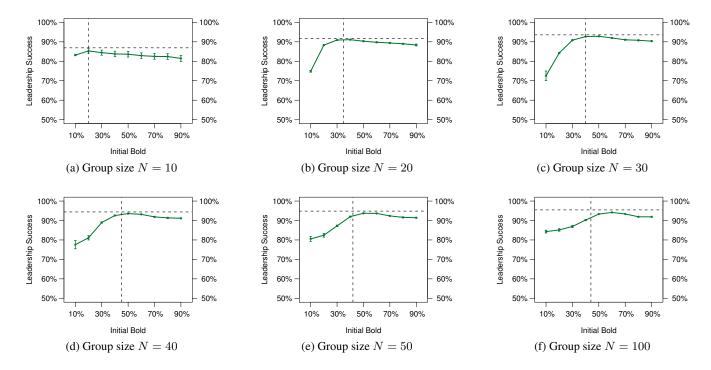


Figure 3: Plots of the percentage of bold individuals in the group versus the mean percentage of successful initiations for different group sizes in simulation treatments with *variable* personalities and fixed *initial* distributions are shown (mean/SD). Error bars representing the standard deviation are also plotted, but are often too small to see. Horizontal and vertical lines denote the "optimal" percentage of bold individuals and the resulting mean success, respectively.

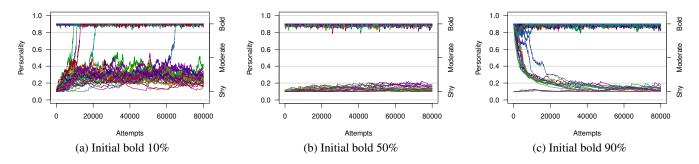


Figure 4: Representative histories of individual variable personalities and the resulting distribution of personalities are shown for a group size of N=40 and three different initial personality distributions.

Discussion

These results allow us to identify three significant contributions. First, as has been observed both in natural systems (Sih and Watters, 2005; Modlmeier and Foitzik, 2011) and in previous research using artificial systems (Pinter-Wollman, 2012), groups containing a combination of bold and shy personalities performed statistically significantly better than groups consisting of a single personality type (see Figure 1). Although the success rate of movement initiations varied with group composition, with some distribu-

tions having significantly higher success than others, these results indicate that there is a broad range of effective distributions of bold and shy personalities that have comparably high performance. However, note that these effective distributions do not include moderate personalities, as all individuals were either bold or shy. While this is consistent with observations in nature which indicate that larger differences between bold and shy personalities result in increased success (Pruitt and Keiser, 2014), the lack of moderate personalities is most likely the result of the personality update rule. For a moderate personality to be stable, it must succeed

as often as it fails. However, this is unlikely since a bolder than average personality is more likely to succeed and become bolder, while a shyer than average personality is more likely to fail and become shyer.

Second, as a result of this broad range of effective distributions, we conclude that the high performing personality distributions are stable equilibriums between bold and shy personalities, much like other stable distributions observed in nature (Mottley and Giraldeau, 2000). This conclusion is further reinforced by simulations using variable personality in which initially extreme personality distributions (e.g., 10% or 90% bold personalities) converged to higher performing distributions (see Figure 2).

One particularly interesting result is that the final distribution of personalities was dependent on the initial distribution. If the final distribution of bold and shy personalities was independent of the initial distribution, one would expect the plots in Figure 2 to be relatively horizontal lines. This is most likely due to the broad range of effective distributions shown in Figure 1 and the lack of sufficient pressure to reach the optimal distribution. Furthermore, when one considers that either consistent initiation success or failure is required to transition a personality from shy to bold or bold to shy, achieving the optimal distribution of personalities appears even less likely.

Lastly, it was surprising that a relatively large percentage of bold leaders (approximately 70%) was supported in many groups. At first glance, this result is counterintuitive given the effects of bold and shy personalities on the collective movement model and the general perception that such a percentage is "too high." However, these results indicate that a bold leader only needs to recruit a sufficient number of shy followers to prevent a quick cancellation of the movement. Once a critical number of followers had joined the movement, other bold individuals joined the movement before the initiator canceled. In the bigger picture, a large percentage of bold leaders can promote higher group success. With more bold leaders, it is more likely that a bold individual initiates first, thus increasing the chances of a successful movement. Given that there are observations of animal groups with both lower (Harcourt et al., 2009b) and higher (Magnhagen, 2006; Sinn et al., 2008) percentages of bold individuals, no general conclusions regarding the relative frequencies of each personality type can be drawn. Current simulations allow for multiple initiations in a group, which may affect the percentage of bold leaders supported in a group.

Conclusions and Future Work

As discussed above, there are three significant conclusions that can be drawn from this work. First, these results show that groups with personality distributions containing a combination of bold and shy personalities perform at a higher level than those having a single personality type. In many

cases, groups with a combination of personalities had almost twice the success in initiating collective movements as single personality groups. Second, these high performing personality distributions were stable equilibriums, as is seen in the simulations with variable personality when initially extreme distributions converge towards the higher performing personality distributions after relatively few simulations. Finally, these results show there is a wide range of effective personality distributions, with some groups supporting up to 70% of the group being bold leaders.

This work represents but one aspect of a larger research effort. The most significant opportunity for future work is to incorporate these results into simulations involving actual movement. Such simulations would provide a number of real-world constraints that may affect the personality distributions discussed in this work, including such things as local communication, dynamic communication networks, and conflicts of interest. Another avenue of investigation would be into the lack of moderate personalities in the results. Stable, moderate personalities are observed in natural systems, but in these simulations, personalities diverge into distinctly extreme bold and shy personalities. Possible reasons for this include the personality function used (see Equation 5) or the collective movement model itself.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. BCS-1124837.

References

- Bernstein, C., Kacelnik, A., and Krebs, J. (1988). Individual decisions and the distribution of predators in a patchy environment. *The Journal of Animal Ecology*, 57(3):1007–1026.
- Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2013). Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41.
- Cote, J., Fogarty, S., Brodin, T., Weinersmith, K., and Sih, A. (2011). Personality-dependent dispersal in the invasive mosquitofish: group composition matters. *Proceedings of the Royal Society B: Biological Sciences*, 278(1712):1670–1678.
- Couzin, I. D., Krause, J., Franks, N. R., and Levin, S. A. (2005). Effective leadership and decision-making in animal groups on the move. *Nature*, 433(7025):513–516.
- Dall, S. R., Houston, A. I., and McNamara, J. M. (2004). The behavioural ecology of personality: consistent individual differences from an adaptive perspective. *Ecology Letters*, 7(8):734–739.
- David, M., Auclair, Y., and Cézilly, F. (2011). Personality predicts social dominance in female zebra finches, *Taeniopygia gut-tata*, in a feeding context. *Animal Behaviour*, 81(1):219–224.
- Dyer, J. R., Croft, D. P., Morrell, L. J., and Krause, J. (2009). Shoal composition determines foraging success in the guppy. *Be-havioral Ecology*, 20(1):165–171.

- Eskridge, B. E. (2012). Effects of local communication and topology on collective movement initiation. In *International Conference on the Simulation and Synthesis of Living Things*, volume 13, pages 155–162.
- Eskridge, B. E., Valle, E., and Schlupp, I. (2013). Using experience to promote the emergence of leaders and followers. In *European Conference on Complex Systems*. Complex Systems Society.
- Fawcett, T. W. and Johnstone, R. A. (2010). Learning your own strength: winner and loser effects should change with age and experience. *Proceedings of the Royal Society B: Biological Sciences*, 277(1686):1427–1434.
- Frost, A., Winrow-Giffen, A., Ashley, P., and Sneddon, L. (2007). Plasticity in animal personality traits: does prior experience alter the degree of boldness? *Proceedings of the Royal Society B: Biological Sciences*, 274(1608):333–339.
- Gautrais, J. (2010). The hidden variables of leadership. *Behavioural Processes*, 84(3):664–667.
- Harcourt, J. L., Ang, T. Z., Sweetman, G., Johnstone, R. A., and Manica, A. (2009a). Social feedback and the emergence of leaders and followers. *Current Biology*, 19(3):248–252.
- Harcourt, J. L., Sweetman, G., Johnstone, R. A., and Manica, A. (2009b). Personality counts: the effect of boldness on shoal choice in three-spined sticklebacks. *Animal Behaviour*, 77(6):1501–1505.
- Hsu, Y., Earley, R. L., and Wolf, L. L. (2006). Modulation of aggressive behaviour by fighting experience: mechanisms and contest outcomes. *Biological Reviews*, 81(1):33–74.
- Katsnelson, E., Motro, U., Feldman, M., and Lotem, A. (2012). Evolution of learned strategy choice in a frequency-dependent game. *Proceedings of the Royal Society B: Biological Sciences*, 279(1731):1176–1184.
- Kurvers, R., Eijkelenkamp, B., van Oers, K., van Lith, B., van Wieren, S., Ydenberg, R., and Prins, H. (2009). Personality differences explain leadership in barnacle geese. *Animal Behaviour*, 78(2):447–453.
- Kurvers, R. H., Adamczyk, V. M., van Wieren, S. E., and Prins, H. H. (2011). The effect of boldness on decision-making in barnacle geese is group-size-dependent. *Proceedings of the Royal Society B: Biological Sciences*, 278(1714):2018–2024.
- Magnhagen, C. (2006). Risk-taking behaviour in foraging youngof-the-year perch varies with population size structure. *Oecologia*, 147(4):734–743.
- Magnhagen, C. and Staffan, F. (2005). Is boldness affected by group composition in young-of-the-year perch (*Perca fluvi-atilis*)? Behavioral Ecology and Sociobiology, 57(3):295–303.
- Modlmeier, A. P. and Foitzik, S. (2011). Productivity increases with variation in aggression among group members in *Temnothorax* ants. *Behavioral Ecology*, 22(5):1026–1032.
- Modlmeier, A. P., Liebmann, J. E., and Foitzik, S. (2012). Diverse societies are more productive: a lesson from ants. *Proceedings of the Royal Society B: Biological Sciences*, 279(1736):2142–2150.

- Montiglio, P.-O., Ferrari, C., and Réale, D. (2013). Social niche specialization under constraints: personality, social interactions and environmental heterogeneity. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 368(1618).
- Mottley, K. and Giraldeau, L.-A. (2000). Experimental evidence that group foragers can converge on predicted producer–scrounger equilibria. *Animal Behaviour*, 60(3):341–350.
- Nonacs, P. and Kapheim, K. (2007). Social heterosis and the maintenance of genetic diversity. *Journal of Evolutionary Biology*, 20(6):2253–2265.
- Oosten, J. E., Magnhagen, C., and Hemelrijk, C. K. (2010). Boldness by habituation and social interactions: a model. *Behavioral Ecology and Sociobiology*, 64(5):793–802.
- Petit, O., Gautrais, J., Leca, J.-B., Theraulaz, G., and Deneubourg, J.-L. (2009). Collective decision-making in white-faced capuchin monkeys. *Proceedings of the Royal Society B: Biological Sciences*, 276(1672):3495–3503.
- Pillot, M.-H., Gautrais, J., Arrufat, P., Couzin, I. D., Bon, R., and Deneubourg, J.-L. (2011). Scalable rules for coherent group motion in a gregarious vertebrate. *PLoS ONE*, 6(1):e14487.
- Pinter-Wollman, N. (2012). Personality in social insects: How does worker personality determine colony personality? *Current Zoology*, 58(4):579–587.
- Pruitt, J. N., Grinsted, L., and Settepani, V. (2013). Linking levels of personality: personalities of the 'average' and 'most extreme' group members predict colony-level personality. *Ani*mal Behaviour, 86(2):391–399.
- Pruitt, J. N. and Keiser, C. N. (2014). The personality types of key catalytic individuals shape colonies' collective behaviour and success. *Animal Behaviour*, 93:87–95.
- Pyritz, L., King, A., Sueur, C., and Fichtel, C. (2011). Reaching a consensus: Terminology and concepts used in coordination and decision-making research. *International Journal of Primatology*, 32(6):1268–1278.
- Şahin, E. (2005). Swarm robotics: From sources of inspiration to domains of application. In Swarm robotics, volume 3342 of Lecture Notes in Computer Science, pages 10–20. Springer.
- Sih, A., Bell, A., and Johnson, J. C. (2004). Behavioral syndromes: an ecological and evolutionary overview. *Trends in Ecology & Evolution*, 19(7):372–378.
- Sih, A. and Watters, J. V. (2005). The mix matters: behavioural types and group dynamics in water striders. *Behaviour*, 142:1423–1437.
- Sinn, D. L., Gosling, S. D., and Moltschaniwskyj, N. A. (2008). Development of shy/bold behaviour in squid: context-specific phenotypes associated with developmental plasticity. *Animal Behaviour*, 75(2):433–442.
- Sutton, R. S. and Barto, A. G. (1998). Reinforcement Learning: An Introduction. MIT Press.
- Wolf, M., van Doorn, G. S., Leimar, O., and Weissing, F. J. (2007). Life-history trade-offs favour the evolution of animal personalities. *Nature*, 447(7144):581–584.

Evaluating Topological Models of Neuromodulation in Polyworld

Jason Yoder¹ Larry Yaeger^{1,2}

¹School of Informatics & Computing, Indiana University ²Google jasoyode@indiana.edu, larryy@indiana.edu

Abstract

Neuromodulation or the modification of neural activity can enhance the biological capabilities of organisms for learning and adaptation (Doya, 2002). Computational models of neuromodulation have demonstrated an advantage for specific tasks (McHale and Husbands, 2003; Parussel, 2006; Soltoggio et al., 2008). In this paper we introduce simple, topological models of neuromodulation and provide a preliminary evaluation of their performance in a foraging task implemented in Polyworld. This work is novel in its evaluation of neuromodulatory models on tasks involving multiple agents competing in an ecologically demanding environment. Polyworld abstracts away the complexity of basic locomotion and digestion, but is biologically consistent in energy conservation, and simulations routinely involve populations of hundreds of neurally controlled agents. The work is also novel in its evaluation of a diverse mixture of models, employing two distinct neural growth models and two distinct models of neuromodulation to draw distinctions otherwise difficult to ascertain. The models developed in Polyworld are explained, evaluated and compared with other computational models of neuromodulation. In particular we find that neuromodulation may be able to enhance or diminish foraging performance in a competitive, dynamic environment, depending upon the nature of the action of neuromodulation and the distribution of neuromodulatory sources.

Neuromodulation

The term neurotransmitter refers to a molecule (usually an amine or neuropeptide) which functions to transmit an activation signal directly. Here, "directly" means from a specific neural component (i.e. one neuron) to a target (another neuron) at a specific time. Neuromodulators, in contrast, generally affect a larger region of neural tissue for a longer time period. In addition, neuromodulators can impact neural activity in ways other than transmitting an activation signal. Despite this semantic difference, certain types of molecules may function both as neurotransmitters and as neuromodulators, in different parts of the same nervous system. The terms are defined functionally, so a molecule can potentially function as one, the other, or both depending upon the context. In fact, these molecules have a rich history in evolution, serving diverse functions across kingdoms (Roshchina, 2010).

Neuromodulators are stored in, and released from, vesicles inside cells. Upon their release they diffuse across tissue in time and space. Brezina (2010) explains, there are "seemingly

infinite" amino acid sequences that make up neuropeptides and this versatility helps explain their flexibility. Brezina summarizes decades of research by suggesting that every aspect of neural activity and any neural process can be modulated. This holds for all levels of the nervous system. In fact, a neural wiring diagram (i.e. synaptic connections between neurons) alone is insufficient to understand most natural neural circuits (Brezina, 2010; Marder, 2012). Progress in understanding neuromodulation will likely require modeling and recording electrical and chemical activity from many parts of a circuit. According to Marder (2012), computational methods will be necessary to observe simulated behaviors emerging from functional properties at a lower level, because we have an incomplete understanding of circuit dynamics and it is simply infeasible to record all of the chemical and electrical activity that occurs in vivo.

Models of Neuromodulation

Computational models of neuromodulation have demonstrated advantages in certain artificial neural network tasks. Parussel (2006) showed that neuromodulation can increase robustness to sensory noise and can improve an agent's ability to adapt when placed in a more complex environment. Soltoggio et al. (2008) showed how neuromodulation of synaptic plasticity could improve performance on tasks involving distant rewards with sparse learning events. Work with GasNets, a type of model which implements neurons emitting virtual neuromodulatory gas clouds (McHale and Husbands, 2003), has shown how certain locomotive control tasks can be better performed by central pattern generators that make use of neuromodulation (McHale and Husbands, 2004). GasNets have been applied to a variety of other tasks, including robotic visual discrimination (Husbands et al., 2010).

However, in all of these projects, despite the dynamics of the testing environments, individual agents have been evaluated in isolation, rather than in competition. In a biological setting, organisms must compete for survival. The competition to survive and reproduce is an important aspect of the fitness landscape. Providing a testing environment that includes these factors may help identify some of the roles that neuromodulation plays in the behavior and evolution of multiple agents. We will compare and contrast these past models with the new project introduced in this paper in greater depth.

Polyworld

Polyworld provides a testing framework to simulate a competitive multi-agent environment with ecological validity. The possibility for action selection in Polyworld is extensive and interesting emergent behaviors have been described (Yaeger, 1994). By incorporating neuromodulation into the neural models of agents in Polyworld we have an opportunity to investigate the roles that neuromodulation could play in behavior and evolution in a competitive environment.

The agents in Polyworld reside in a two-dimensional world, with edges that the agents cannot move beyond. Within this world food is placed in clusters and may be relocated frequently in order to require agents to evolve foraging behaviors for survival. The agents in Polyworld are controlled by artificial neural networks (ANNs). Each agent receives a one-dimensional vector of red, green, and blue visual input from a forward facing sensor, which feeds into the input layer of neurons in its ANN. The agents can control the focus of their vision, i.e. select the width of their field of vision. There are red, green, and blue components of color in the agents' bodies and in the environment, with all food appearing green. The intensity of an agent's red color component is linked to the activation level of its fight neuron/behavior, and the intensity of its blue color is linked to the activation level of its mating neuron/behavior. The input neurons are activated with strengths based upon the visual input received. The input neurons are connected to hidden neurons, which may in turn be connected to and from other hidden neurons, before ultimately connecting to output neurons. The different behaviors that agents can exhibit (moving, turning, eating, fighting, mating, focusing and lighting) are controlled by the activation of these output neurons.

By assigning primitive behaviors to single output neurons instead of complex neural circuits, agent's actions are abstracted and minimal neurons are devoted to simple actions. This allows more neurons to be devoted to action selection. An agent expends energy by performing these actions (it also expends a fixed amount of energy over time) and it "dies" if it runs out of energy. An agent's energy increases when it "eats" at a location with food. Agents can fight and when doing so they damage a separate, stored health value, which results in the agent's death upon reaching zero (health, like energy, is expended by actions and replenished by eating). Agents killed leave behind food equivalent to what was left of their energy (some of which may remain, even though health is zero). Each agent has a genetic code representing the connectivity of its neural network (but not synaptic weights), as well as size, speed, and other traits. For a full catalogue of genes and more details about the metabolism and physiology of agents see (Yaeger, 1994).

Polyworld simulations operate in discrete time-steps. At each time-step, the ANN of each agent is updated. An agent's current orientation and the state of the world determine what it "sees." The synaptic connections then propagate activation. Each target neuron's activations are summed. Based on these summations each pre-synaptic neuron (input or hidden) transmits the appropriate activation to its post-synaptic neurons. Output neuron activations determine the agent's behavior(s) at each time-step. The synaptic weights in the ANNs are updated according to Hebbian plasticity, (i.e. when

neurons fire together, the synaptic weights between them increase). This unsupervised learning is well-suited to this level of complexity, and has stronger biological feasibility than, for example, back-propagation of error.

If two agents simultaneously activate their *mating* neuron sufficiently when they are physically close, they will produce an offspring. The genes from the two parent agents are crossed over to generate the genes for the offspring; see (Yaeger, 1994) for details. When the number of agents in a simulation gets critically low (as a result of more death than reproduction) agents can be generated using an online genetic algorithm, however for these simulations a more graduated and less invasive technique is used in which energy requirements on the agents are relaxed as the population falls, guaranteeing the population never goes below a minimum. (Though simulations can still fail, with populations reaching this minimum and never recovering.)

It is important to emphasize that agents do not pass on acquired "knowledge" to offspring. The neural network's connectivity is passed on genetically, but the synaptic weights are randomized at birth. Inspired by Linsker's work on visual cortex simulation (Linsker, 1988), Polyworld relies on network structure and Hebbian learning to produce evolutionarily useful behaviors. Because synaptic weights are not stored in the genome of agents, each agent must learn during its lifespan. This helps minimize the genome and also establishes a non-Lamarckian evolutionary model. Thus, the challenge of survival and reproduction for agents is significant; they cannot simply memorize food cluster locations and pass them on to their offspring. Instead, they must evolve a neural structure that learns to find enough food in order to survive long enough to find another agent to mate with, otherwise they will "die" and exit the gene pool.

Polyworld may have impassable barriers placed in an environment. These structures visually obscure anything behind them and require agents to navigate around them. These may be used to increase the difficulty of survival for agents and will be discussed more when dealing with the testing procedure.

Neural Architectures

There are two neural architectures—two methods for growing neural topologies—in Polyworld. In the first architecture, the neurons are non-spatial; i.e., neurons' relationships with other neurons are determined purely topologically. Despite lacking spatial properties, the neurons are clustered into groups and many properties of neurons are set at the group level. This neural architecture will henceforth be referred to as the *groups* architecture, because neural connectivity is specified at the group level. The synaptic connection density from one group to another is controlled genetically.

The second architecture, henceforth referred to as the *sheets* architecture, includes a spatial component. Each neuron is located within a 3-dimensional virtual space. Similar to the *groups* architecture, there are clusters of neurons organized together. However, in this case they are all located in a single plane or sheet. All the input neurons are located at one edge (the front) of the space. The output neurons are located at the opposite edge (the back) of this space. In between are sheets of hidden neurons. These sheets of neurons connect to one another via synapses. Synapses are probabilistically assigned

from neurons on a source sheet within a radius of the potentially offset projection of a target neuron (Figure 1). This receptive field of the target neuron determines which neurons are connected to it from the source sheet. This use of probabilistic receptive fields was done to mirror biology (Mitchison, 1991; Murre, 1995; Bullmore and Sporns, 2009) and help bias towards small world networks (Berry and Temam, 2007).

The performance of agents with and without neuromodulatory capabilities was compared using these two different neural architectures. In the next section we explain the simple model of neuromodulation that was incorporated into these neural architectures.

Properties Modulated

In this paper we report on work incorporating neuromodulation into both the *sheets* and *groups* architecture. This model is simple and exploratory and while we do not make any effort to model diffusion, we do attempt to model the neuromodulation of two types of neural activity. One property that can be modulated is the sum total of activations from pre-synaptic neurons to a given post-synaptic neuron. Alternatively, the synaptic plasticity of a neuron, i.e. the rate at which pre-synaptic weights change, can be modulated. In our model either of these properties can be increased or decreased via neuromodulation.

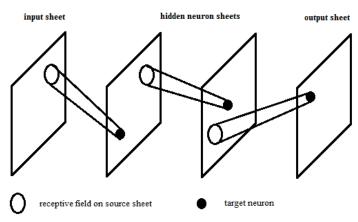


Figure 1. Each neuron has a probability of growing synapses from a receptive field on another sheet. The center of the receptive field on the source sheet may be offset relative to the target neuron. The hidden neurons are located on sheets between the input and output sheets.

Model of Neuromodulation

A special modulatory neuron type was added to the preexisting excitatory and inhibitory neuron types. The special modulatory neurons have synaptic connections following standard connectivity rules. However, when activated, their effect on target neurons differs from standard neurons. They either influence the synaptic plasticity (learning rate) of the target neuron's pre-synaptic connections, or the overall level of activation from all pre-synaptic connections. These modulations take place at each time-step, and hence the modulation takes place at the same speed as the regular neural activity. The model of neural activation for the standard excitatory and inhibitory neurons is given by

$$x_i = \sum a_j^t s_{ij}^t$$

$$a_i^{t+1} = 1 / (1 + e^{-\alpha x}i)$$

Yaeger (1994) explains:

where a_i^t is the neuronal activation of neuron j at time t (the beginning of this time step), s_{ij}^t is the synaptic efficacy from neuron j to neuron i at time t, a_i^{t+1} is the neuronal activation of neuron i at time t+1 (the end of this time step), and α is a specifiable logistic slope.

Modulation of neural activation changes the equations to be:

$$\begin{split} x_i &= m_i * \sum a_j^t \, s_{ij}^t \\ a_i^{t+1} &= 1 \, / \, (1 + e^{-\alpha x} i \,) \\ m_i &= \begin{cases} 1, & M = 0 \\ \frac{\sum a m_j^t * s m_{ij}^t}{M}, & M \neq 0 \end{cases} \end{split}$$

Where am_j^t represents the neuronal activation of a modulatory neuron j at time t and sm_{ij}^t represents the synaptic efficacy from neuron j (modulatory) to neuron i at time t. M represents the number of modulatory neurons with connections to neuron i. The division by M was done to take the average modulation value across all the modulatory neurons that are influencing the target neuron, rather than the sum of those values. This was done to normalize modulatory effects so that the number of modulatory connections does not change the degree of the modulation.

The model of synaptic plasticity for the standard excitatory and inhibitory neurons is given by

$$s_{ij}^{t+1} = s_{ij}^{t} + \eta_{kl}^{c} (a_i^{t}+1 - 0.5) (a_j^{t} - 0.5)$$

Again, Yaeger (1994) explains:

where s_{ij}^{t+1} is the synaptic efficacy from neuron j to neuron i at time t+1, and η^c_{kl} is the learning rate for connections of type c (e-e, e-i, i-i, or i-e) from group l to group k. An optional multiplicative decay term may also be applied to the synaptic efficacy.

Modulation of synaptic plasticity changes the equations to be:

$$s_{ij}^{t+1} = s_{ij}^{t} + \eta^{c}_{kl} (a_{i}^{t} + 1 - 0.5) (a_{j}^{t} - 0.5) * m_{i}$$

$$m_{i} = \begin{cases} 1, & M = 0 \\ \sum am_{i}^{t} * sm_{ij}^{t}, & M \neq 0 \end{cases}$$

Where am_j^t again represents the neuronal activation of a modulatory neuron j at time t and sm_{ij}^t represents the synaptic efficacy from neuron j (modulatory) to neuron i at time t. M again represents the number of modulatory neurons with connections to neuron i. Also, the c in η^c_{kl} can now refer to connections of types e-e, e-i, i-i, i-e, m-e, m-i, m-m, e-m, and i-m. Again, the division by M was done to take the average modulation value across all the modulatory neurons that are influencing the target neuron.

Task

In simulations, Polyworld's environment was configured to generate a patch of food in one of three locations. The patch was relocated, cyclically every 2000 steps. After the first 10,000 steps of a simulation, dynamic, impassable barriers were added in the environment. After starting at size zero, these barriers were programmed to grow slowly when the population was doing well (above 175 agents). When a population was struggling (below 120 agents), the barriers started to shrink quickly. If the population was performing poorly (below 90 agents), barriers would shrink twice as fast. Figure 2 demonstrates the layout of the possible food patches (green squares) and three snapshots of potential barrier sizes.

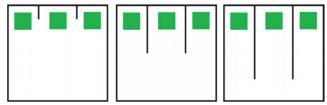


Figure 2. Left: agents require minimal navigation; food locations are mostly within line-of-sight. Middle: barriers obstruct view and navigation. Right: barriers extend far, obstructing agents significantly.

The task of foraging in order to find food in a new location becomes more difficult as the barriers increase in size. This should be intuitive when examining Figure 2. The reasons for this increased difficulty are two-fold. First, the agents must travel a greater distance (expending additional energy) to get from one food patch to the next when barriers are present. The longer the barrier the greater the shortest distance from one food patch to another will be. Secondly, when a food patch is relocated, agents near the old food patch will have their vision of the new patch obscured by the barriers. This forces agents to explore for food (expending additional energy), and simply mapping visual inputs to actions is no longer an adequate foraging strategy. At their maximum, barriers could reach 90% of the way from one edge of the world to the other.

Monitoring how these barriers grow and shrink over time can give an estimate of how effective a given population of agents is at adapting to the environment. Population level increases and decreases are very frequent in simulations, though as long as the population stays above a certain threshold we consider the performance to be acceptably good. When the population falls below the acceptable level, we know it is struggling to sustain itself. However, due to general fluctuations in population level, a low-fitness population may have a temporarily high population of agents, or a high-fitness population may have a temporarily low population. To compensate for this and prevent complete population crashes, barriers grow slowly and shrink quickly, thus providing a better match of world difficulty to agent population over time.

Highly fit populations will increase in size, and poorly fit populations will decrease in size over time. If a population remains consistently large, even as patches relocate, then the barrier sizes will gradually increase. Examining the size of the barriers at a given time is therefore one means of measuring the fitness of the agents at that time, since barriers will only grow with a consistently sustained large population. We

investigated the use of raw barrier size as a measure of agent performance, but due to relatively rapid fluctuations in population and corresponding fluctuations in barrier size, the measure proved to be very noisy. We also investigated using just a "high water mark" of barrier size, but while less noisy, this measure failed to capture intermediate levels of population recovery and barrier growth, ignoring near die-offs and substantial differences in population recovery times from such events.

We were able to obtain a much smoother and more representative measure of the performance of a population of agents by integrating barrier size over time. A simple running summation of barrier size defines a metric that increases more rapidly when populations perform well, thus allowing the barrier to grow quickly and consistently (or at least rebound frequently). Conversely, the metric grows more slowly when poorly performing populations force the barrier size to shrink. We call this metric *Productivity* and use it to assess the performance of each population of agents.

It is also worth noting that this approach of growing and shrinking barriers helps guide evolution toward increased foraging ability. As the barriers increase in size, the challenge of foraging increases and those that adapt best to the increased challenges will pass on their genes. This can be thought of as breaking a challenging problem into a series of less challenging sub-problems that can be solved in steps, yielding a continuous reshaping of the fitness landscape. Similar results could likely be obtained without this dynamic fitness shaping, but we would expect only a small fraction of seed populations to succeed in a world that starts off at maximum difficulty. Use of a dynamically shaped fitness landscape allows every simulation to succeed to one degree or another, depending upon the Productivity of its population of agents, and the degree of success then serves as our fitness metric rather than the fraction of successful populations. This is vastly more efficient than an un-shaped environment in the use of computational resources.

A protocol was implemented to compare simulations of agents including and excluding neuromodulatory features. There are two neural architecture variants (*Groups* and *Sheets*) each with the possibility of a standard ANN with no neuromodulation (*Base*), one including neuromodulation of neural activation (*Activation*), or one including neuromodulation of synaptic plasticity (*Plasticity*), generating six possible agent architectures to compare: *GroupsBase*, *GroupsActivation*, *GroupsPlasticity*, *SheetsBase*, *SheetsActivation*, and *SheetsPlasticity*.

Ten runs of 200,000 time-steps were run for variants of the six agent architectures. The only difference in the simulations aside from the specified neural architecture was a pseudorandom number generator seed used for randomizing many stochastic features of the simulation. The Productivity of the agent population in each simulation was calculated every 100 time-steps and used to assess the performance of each variant.

Results

Figures 3 through 7 display plots of agent Productivity versus time. In each plot there are two curves, each showing mean Productivity, with standard-error bars, for ten simulations each of two different architectures. In addition, a

plot of statistical significance (1 - p-value) is displayed at the bottom as a finely dashed line, corresponding to any differences in mean Productivity between the differing neural architectures at each time-step. The p-values were calculated using an independent Welch's t-test (a Student's t-test allowing unequal variance).

Looking at significance at each time step could run the risk of random statistical aberrations suggesting significance where there is none, but the samples at adjacent time steps are not independent here, and the smoothness of these curves means that integrating over longer time scales would produce comparable levels of significance (within modest time windows). The Productivity metric being analyzed is also itself a summed statistic. So for multiple reasons a repeated sampling correction is not required. We also present wholerun significance results in Figure 8, which are consistent with the intuitions gained from the temporal traces of significance.

To aid understanding, colors and line styles are consistently used in the following manner: Black indicates a standard ANN without neuromodulation (Base), blue indicates

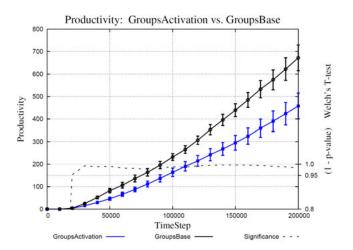


Figure 3: Activation neuromodulation compared to Base (no neuromodulation) using the Groups architecture.

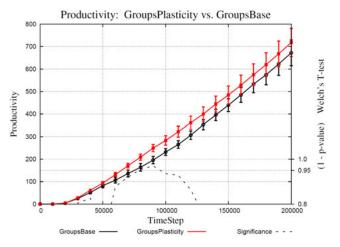


Figure 4: Plasticity neuromodulation compared to Base (no neuromodulation) using the Sheets architecture.

modulated neural activation (Activation), and red indicates modulated synaptic plasticity (Plasticity). Results obtained with the Groups neural architecture are displayed using a solid line; those using the Sheets architecture are shown with a dashed line. A dot-dash line style is used in Figure 7 when the Groups and Sheets results are aggregated.

Figures 3 and 4 compare the Activation and Plasticity neuromodulation results, respectively, to the Base (no neuromodulation) results using the Groups architecture.

Figures 5 and 6 compare the two neuromodulation results to the Base results using the Sheets architecture.

Figure 7 compares Activation neuromodulation to Plasticity neuromodulation using aggregated Groups and Sheets results.

Figure 8 presents a whole-run integrated Barrier Size comparison between all three neuromodulation models (Base, Activation, and Plasticity) for both neural architectures (Groups and Sheets). As Productivity was defined to be integrated Barrier Size, this ends up being equivalent to a comparison of the final time-step's Productivity. The box and whiskers plots are standard, showing median, quartiles, and

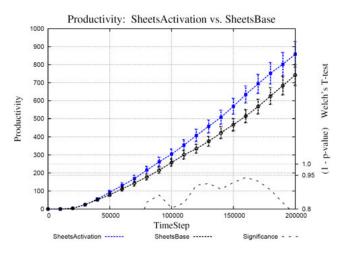


Figure 5: Activation neuromodulation compared to Base (no neuromodulation) using the Sheets architecture.

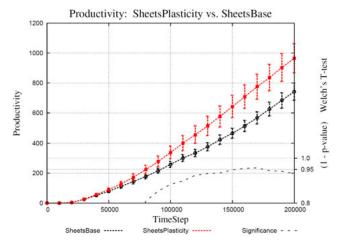


Figure 6: Plasticity neuromodulation compared to Base (no neuromodulation) using the Sheets architecture.

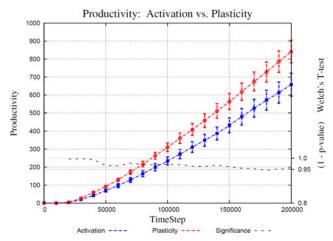


Figure 7: Activation neuromodulation compared to Plasticity neuromodulation for combined Groups and Sheets results.

extrema. Overlaid on the centerline of the box and whiskers are the individual values from the ten runs for that model and architecture. Adjacent to the box and whiskers are representations of the mean and 95% confidence intervals, so that significance can be inferred directly from the plot.

Discussion and Conclusions

Comparisons amongst our models of neuromodulation provide some results that are statistically significant and some that are merely suggestive.

In what initially we were concerned might be an anomalous result, Figure 3 shows a highly statistically significant difference between Activation neuromodulation and the Base model (with no neuromodulation). For this particular neural neuromodulatory model and architecture, neuromodulation significantly reduces the Productivity of agents compared to those using no neuromodulation. This is consistent with the reduced performance exhibited by agents using activation modulation in the absence of noise in the Curiously, work of Parussel (2006).neuromodulation seems to have a small positive effect on agent Productivity when instantiated in the Sheets neural architecture (see Figure 5). We hypothesize that this reversal of effect may be the result of substantially different distributions of modulatory neurons between the Groups and Sheets architectures. In the Groups neural growth model, modulatory neurons are specified and linked into the neural graph as a group, producing topological clusters of modulatory neurons, whereas in the Sheets architecture, due to the spatially distributed growth model, the modulatory neurons tend to be more evenly distributed amongst the excitatory and inhibitory neurons, at least initially. More work is required to ascertain whether this is actually the source of the discrepancy, but if correct, this suggests one possible hypothesis regarding the distribution and possibly the growth of modulatory neurons in biological brains.

Figure 4 shows there is little statistically significant difference between Plasticity neuromodulation and a lack of neuromodulation, but the brief excursion of significance above 0.95 and the consistently higher mean Productivity with

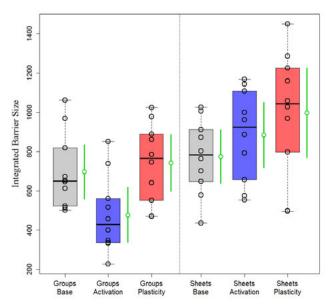


Figure 8: Whole-run Barrier Size / Final Productivity comparison between Base, Activation, and Plasticity neuromodulation using Groups and Sheets architectures. See text for explanation of plot elements.

Plasticity neuromodulation are vaguely suggestive of a difference that might show up better with more data or a more challenging problem (or at least a problem more suited to the expected benefits of neuromodulation; see the Future Directions section).

Figure 5 again compares Productivity between Activation neuromodulation and the Base (no modulation), this time for the Sheets neural architecture. Though the results fall short of statistical significance we can see that the significance curve almost reaches 0.95 and stays somewhat elevated for a moderately long time. The magnitude of the separation between the neuromodulation and Base curves is also larger than that between Plasticity and Base with the Groups architecture, and, as mentioned previously, in the opposite direction of that between Activation and Base with the Groups architecture. So once again, there is a hint that there might be something to go after here, despite the lack of statistical significance.

The Plasticity in Sheets results of Figure 6 are a bit stronger, with significance rising above 0.95 for on the order of 20,000 time steps, though significance is falling towards the end of these simulations. The magnitude of the separation between these curves is also the largest of any of the pairs of results, with Plasticity neuromodulation providing an apparent benefit to agents' Productivity. It is impossible to say whether significance will continue to diminish or grow again if and when we extend these runs to longer times, but this is the most positive individual result of this series of simulations. (The negative being the impact of Activation neuromodulation on the Groups architecture.)

Figure 7 gets at this difference between Activation and Plasticity models of neuromodulation, aggregating the Groups and Sheets results. These results are difficult to interpret, despite fairly extensive statistical significance, because the

differences are largely dominated by the negative results of the GroupsActivation model. But even with the Sheets architecture alone, Plasticity performs better than Activation, if not to a statistically significant degree, so far.

Figure 8 recapitulates and encapsulates all of these results, showing a whole-run integrated Barrier Size comparison of all neuromodulation models (Base, Activation, Plasticity) for both neural architectures (Groups, Sheets). The box and whiskers elements give a good feel for the overall spread of the data, and the mean and 95% confidence intervals show the limited statistical significance, except that resulting from the negative GroupsActivation model. As stated previously, we need to better understand the GroupsActivation results.

The overall trend amongst the different models of neuromodulation in the Sheets architecture offers another hint that there is something worth pursuing here. There is hope that we will be able to distinguish efficacy between these different models with additional effort.

In the end, we feel the results are weak but suggestive. Differences between neuromodulation and neuromodulation were small but consistently positive for the Plasticity model, and significantly negative for the Activation model in the Groups architecture, yet slightly positive in the Sheets architecture. Overall there is a hint that Plasticity modulation may ultimately outperform However, further work is needed to see if any modulation. of these tentative suggestions hold true.

Future Work

These models do not currently account for any differences in time course between standard neural activity and neuromodulatory activity. They also fail to account for any spatial diffusion dynamics. They were designed as exploratory models to assess how neuromodulation of some specific neural activities would influence the ability of a population of competing agents to evolve and perform in a biologically inspired foraging task. Given the modest suggestion of potential here, we intend to extend the models of neuromodulation, adjust the challenges faced by the agent populations, and increase the range of data we gather and examine from simulations.

Specifically, we intend to develop models of spatial diffusion inspired by, but extending, GasNets neuromodulation. We believe the ability to affect neural dynamics on multiple time scales may prove to play a critical role in any benefits conveyed by neuromodulation.

We also intend to explore some behavioral challenges that may be better suited to the benefits conveyed by neuromodulation. It is possible that the slowly growing and shrinking barriers do not represent the kind of short-time-scale cognition and error-driven learning that neuromodulation is thought to serve (O'Reilly et al., 2014), and that faster, more transient behavioral challenges will allow neuromodulation to have a stronger impact on agent fitness. Though interaction between agents is already one of the most important aspects of an agent's environment, we are also considering a more social task through the introduction of stigmergy. Previous work has also demonstrated differences in the value of neuromodulation in the presence or absence of noise (Parussel, 2006), which would be easy to investigate in Polyworld.

The model developed in this paper is similar to specific components of modeling work done by Parussel (2006) and Soltoggio et al. (2007) in addition to work done using GasNets (McHale and Husbands, 2003). The multiplicative modulation of neural activation can be seen as analogous to multiplicatively altering the transfer gain function in the GasNets model, and to the multiplicative modulation of neural activation in Parussel's model. The neuron-to-neuron multiplicative modulation of synaptic plasticity in this model is analogous to the modulation of synaptic plasticity used by Soltoggio. We expect that having multiple models of neuromodulation will allow us to tease apart their differences and develop a better understanding of their strengths and weaknesses.

Similarly, we expect that having two different (Groups and Sheets) neural architectures—two different neural growth models—will be powerful for this exploratory type of study. It is reasonable to think that the computational characteristics of neuromodulation are dependent on neural architectures, including their spatial organization, and having both models will allow us to find similarities and make distinctions not otherwise possible. In particular, the reversal in effect of Activation modulation between the Groups and Sheets architectures needs to be investigated and better understood. We have experience characterizing these evolved networks using graph theoretical techniques (Yaeger et al., 2010; Yaeger, 2013) which we expect will offer additional insights into these behavioral differences, and which hopefully will confirm or refute our hypothesis regarding clustered vs. uniformly distributed modulatory neurons and their effect on Activation modulation.

Initial attempts at running this experiment resulted in much more erratic data, so the barrier growth rate was decreased and the shrink rate was increased. This was done to limit the variance within and between runs. This parameter space could be further explored to help tease apart differences in performance. Alternatives to early modeling decisions, such as averaging versus summing neuromodulation, need to be explored. Furthermore, simply running more simulations could be helpful in identifying trends that were not easily observed with just ten simulations.

One way of phrasing the question that this model and paper seek to investigate is whether the computational characteristics of these special modulatory neurons grant an advantage over networks consisting entirely of standard neurons. In general the number of neurons present in each agent were equally divided between excitatory, inhibitory, and modulatory neurons at the start of evolution. Over time, the number and fraction of modulatory neurons present in agents may increase or decrease, and will be monitored in future work

In this paper we have incorporated a simple exploratory model of neuromodulation into Polyworld and run experiments to ascertain how the availability of special neuromodulatory neurons impacted the evolution and Productivity of a population of competing agents pressured to evolve increasing foraging abilities. Results are tenuous but suggest that some forms of neuromodulation may convey either an evolutionary advantage or disadvantage on agents, with the distinction being based upon neural topology. The strength of any such advantage or disadvantage and the

conditions under which it is maximized are open questions. These are still preliminary results and our hope is that with this project we have laid the ground work for testing more advanced models of neuromodulation and provided ourselves the opportunity to design new and better experiments with different challenges for the agents. We expect that future work will better illuminate the distinctions and benefits of different models of neuromodulation.

Acknowledgements

The authors would like to thank the ALife 14 conference reviewers and its Chair, Hiroki Sayama, for both valid criticism and support, which resulted in a substantially improved version of this paper. We are also grateful to Colin Allen for ongoing discussions and for illuminating some key issues in interpreting agent Productivity and statistical significance in these results.

References

- Berry, H. and Temam, O. (2007). Modeling self-developing biological neural networks. *Neurocomputing*, 70: 2723-2734.
- Brezina, V. (2010). Beyond the wiring diagram: signalling through complex neuromodulator networks. *Philosophical Transactions of the Royal Society. Biological Sciences*, 365(1551): 2363-2374.
- Bullmore, E. and Sporns, O. (2009). Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Review Neuroscience*, 10:186-198.
- Doya, K. (2002). Metalearning and neuromodulation. *Neural Networks*, 12(4-6): 495-506.
- Husbands, P., Philippides, A., Vargas, P., Buckley, C. L., Fine, P., Di Paolo, E., & O'Shea, M. (2010). Spatial, temporal, and modulatory factors affecting GasNet evolvability in a visually guided robotics task. *Complexity*, 16(2), 35-44.
- Linsker, R. Self-organization in a perceptual network. *Computer* 21.3 (1988): 105-117.
- Marder, E. (2012). Neuromodulation of neuronal circuits: back to the future. *Neuron*, 76(1): 1-11.
- McHale, G. and Husbands, P. (2003). GasNets and other Evolvable Neural Networks applied to Bipedal Locomotion. In From Animals to Animats 8: Proceedings of the Eight International Conference on Simulation of Adaptive Behavior (SAB 2004)
- McHale, G. and Husbands, P. (2004). Quadrupedal locomotion: GasNets, CTRNNs and hybrid CTRNN/PNNs compared. In Pollack, J., Bedau, M., Husbands, P., Ikegami, T., and Watson, R. A., editors, Artificial Life IX: Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems, pages 106-113. MIT Press, Cambridge, MA.
- Mitchison, G. (1991). Neuronal branching patterns and the economy of cortical wiring. *Proceedings of the Royal Society B: Biological Sciences*, 245(1313): 151-158.

- Murre, J. M. (1995). The connectivity of the brain: multi-level quantitative analysis. *Biological Cybernetics*, 73(6): 529-545.
- O'Reilly, R. C., Munakata, Y., Frank, M. J., Hazy, T. E., and Contributors (2014). *Computational Cognitive Neuroscience*. Wiki Book, June 1, 2014 Edition. Chapter 4, Learning Mechanisms. URL: http://ccnbook.colorado.edu
- Parussel, K. (2006). A Bottom-up Approach to Emulating Emotions Using Neuromodulation in Agents. PhD. Thesis, Department of Computer Science and Mathematics, University of Stirling.
- Roshchina, V. V. Evolutionary considerations of neurotransmitters in microbial, plant and animal cells. (2010). In M. Lyte and P. P. E. Freestone, editors, Microbial Endocrinology: Interkingdom Signaling in Infectious Disease and Health, pages 17–52. Springer, New York, NY.
- Soltoggio, A., Durr, P., Mattiussi, C., & Floreano, D. (2007, September). Evolving neuromodulatory topologies for reinforcement learning-like problems. In *Evolutionary Computation*, 2007. CEC 2007. IEEE Congress on, pages 2471-2478. IEEE.
- Soltoggio, A., Bullinaria, J. A., Mattiussi, C., Dürr, P., & Floreano, D. (2008). Evolutionary Advantages of Neuromodulated Plasticity in Dynamic, Reward-based Scenarios. In *Artificial Life XI*, pages 569-576. MIT Press, Cambridge, MA.
- Yaeger, L. S. (1994). Computational Genetics, Physiology, Metabolism, Neural Systems, Learning, Vision, and Behavior or Polyworld: Life in a New Context. In Langton, C. G., editor, *Proceedings of the Artificial Life III Conference*, pages 263–298. Addison-Wesley, Reading, MA.
- Yaeger, L., Sporns, O., Williams, S., Shuai, X., & Dougherty, S. (2010). Evolutionary Selection of Network Structure and Function. In Fellermann et al., editors, *Proceedings of the Artificial Life XII Conference*, pages 313-320. MIT Press, Cambridge, MA.
- Yaeger, L. S. (2013). Identifying neural network topologies that foster dynamical complexity. Advances in Complex Systems, 16(02n03).

The Effects of Elitism on Spatial Coevolutionary GAs

Morgan McLaughlin and Mark Wineberg

University of Guelph, Ontario, Canada

Abstract

Elitism is a common concept in the standard Genetic Algorithm (GA), but one that has received little to no research when considered on spatial coevolutionary GAs. This paper attempts to address this by implementing and analyzing elitism mechanisms within a spatial coevolutionary system. To this end we develop a new globally selected local elitism and compare it with the more basic strictly non-local elitism generally seen in standard GAs. To aid in this analysis we also create a new simple function, one-max matching, which incorporates the simplicity of one-max yet is highly multimodal. We find that elitism seems to have a massive positive effect on coevolutionary systems, Heat maps show that the globally selected local elite often cluster within in the spatial network. Finally, surprisingly it took two elite per population in a coevolutionary system to see any benefit from elitism; we don't know why.

Introduction

The Spatial Coevolutionary Genetic Algorithm (Hillis, 1990; Weigand and Sarma, 2004) brings together two paradigms within evolutionary computation: coevolutionary systems and spatially determined mating restrictions. However the spatial coevolutionary system provides the opportunity to introduce a hybrid mechanism not present in either system individually, spatial evaluation, which had been incorporated in both Hillis' and Weigand and Sarma's systems. In coevolution, the evaluation of a member from one population is based on the genotype or phenotype of one or more members from another population. With spatial evaluation the selection of the member, or members, of the other population is restricted to a spatially determined subset. The evaluative spatial structure is usually created such that members from the two populations that have an evaluative edge between them have neighbors that also have evaluative connected edges. Unlike the reproductive spatial structure, the evaluative spatial structure forms a bipartite graph between the populations.

While spatial coevolution has been around as long as coevolution has in the evolutionary algorithm world (both simultaneously introduced in Hillis 1990), only limited analysis has been done on these systems (Pagie and Hogeweg, 1997; Weigand and Sarma, 2004; Williams and Mitchell, 2005; Mitchell et. al. 2006) and none of them investigated the effects that elitism might have in such systems.

Elitism, where the best member from the previous population is copied unchanged into the current population, is a mechanism that has been used in the Genetic Algorithm (GA) almost since its inception (De Jong, 1975). Its purpose is to

prevent a population from losing the best solution it has found up until that point. While the elitism mechanism is not one that maps directly to the natural world of genetics, it has over time become ubiquitous in the GA world because of its effectiveness.

The addition of a spatial structure to reproduction added even more opportunity with the addition of local elitism. First introduced by De Jong and Sarma (1995) to fine-grained parallel GA systems and extended to any spatially structured GA (Sarma and De Jong, 1997), local elitism extends tournament selection, the most common selection technique used for spatial GA systems, to include the population member from the current generation, unaltered by mutation or crossover, in the spatial slot being filled when creating the next generation. With the addition of local elitism, the original "global" elitism becomes unnecessary since if the globally elite member would be the best member in the next generation, it would win its slot in its local tournament selection. If it doesn't win its slot, then it wouldn't be the elite after the generation following, so the effective difference is very short-lived.

Adding elitism to spatial coevolutionary systems is an obvious step to do, but probably to reduce complicating factors in their analyses, it has not been included in any of the spatial coevolutionary systems previously discussed. While elitism is the primary mechanism used in the Potter and De Jong's (1994) cooperative coevolutionary system, where each of the members from a population being evaluated is evaluated against the elite member of the other population, there is no spatial structure used for either the reproductive or evaluative aspects of the algorithm. This is unlike the cooperative coevolutionary system of Weigand and Sarma, 2004, which did use spatial structures but not elitism.

In this paper we will study the effects of elitism within a spatial coevolutionary system demonstrating its importance and utility. To accomplish this we needed to create a technique for elitism that can be used to seamlessly transition between the standard GA's global elitism and the spatial GA's local elitism. To aid in the analysis we also develop a simple test problem based on one-max, but is also multimodal.

In the next section we will define the spatial coevolutionary GA used for our set of experiments. We will then describe the experiments that will be used to determine the effectiveness of elitism in spatial coevolutionary systems. This will be followed by the results of the experiments, analysis and discussion of the results, and the conclusion.

Implementation

Elitism

In a regular GA, elitism is important as it prevents the best solution found so far from being accidently lost through the vagaries of selection. This allows important genetic information to continually be disseminated throughout the population. The effect of the elite is not so clear when dealing with spatial GAs since the genetic material from the elite member can only be of immediate use to its local neighborhood. Local elitism (De Jong and Sarma, 1995) allows for the elite to emerge locally through tournament competition as described previously. However this produces a "ratcheting effect" everywhere, were no solution in the next population can be worse than from the previous population, which is not present in a standard GA. While it is possible to invoke a tournament between parents and children for inclusion in the next generation (which is done in some GA implementations) this is far from being standard, and could easily place too strong a selection pressure on the population. On the other hand, if you treat the "global" elite in a spatial population the same as you would in a standard GA and just copy it unchanged into the next generation, it limits the amount of spatial variation in its locality, which is already limited by spatial reproduction and can easily stall evolutionary progress in the very area that has the most promising solution. Since the effect of elitism is so different between the standard GA and a spatial GA (and even more so a spatial coevolutionary GA), it becomes very difficult to compare elitism directly between the two systems.

To facilitate direct comparison, we introduce an "intermediate" elitism mechanism between the "global" elitism of the standard GA and local elitism of the spatial world. Instead of implementing local elitism everywhere, we choose *k* unique elite and for each of those "global" elite, we apply local elites In other words, we globally select and locally implement.

This globally selected local elitism allows us to smoothly transition between the elitism used in the standard and the spatial GA. When the spatial system is fully connected, the elite member is compared against offspring that come from parents that can be anywhere in the population¹. The chances are therefore that the elite member will win the tournament and be copied into the next generation as with a standard GA. When k is set the full population size, the system becomes local elitism. In our experiments k is set to 1, 2, 10 and 100, where 100 is the full population size.

We will use this version of elitism in all of our experiments except for one. In that experiment we allow the elite member to be copied into the next generation whether or not it won the local tournament. This will allow us to observe whether, as predicted above, the detrimental effects that non-local elitism should have on spatial systems actually transpires.

General GA

One of the main experimental controls we imposed for this study was to use as simple a GA on as simple a problem as possible without becoming overly trivial and providing sufficient features to differentiate between various properties of interest. With this in mind we chose to use a population size of 100, big enough to setup non-trivial spatial networks, yet small enough to be able to study with a fair amount of ease. We have a 0.8 probability of crossing over using uniform crossover with a parameter of 0.3 (i.e. a 0.3 probability of crossing over a bit location). The mutation rate is set to 2/L where L is the chromosome length used in the problem and which is problem specific. For convenience we applied a max generation of 5000 for all of our simulations to ensure that the majority of runs will complete, but if not that they would still terminate in a timely manner; a simulation which has not completed after 5000 generations is simply considered a 'failure', having failed to find the optimal solution. Each experiment will be run through 50 repetitions in order to obtain statistically relevant results.

Spatial GA

The reproductive spatial GA used for coevolutionary system, while keeping close to a standard spatial system, does have some slight differences introduced to allow for a seamless transition between the spatial GA and the standard GA, which is designed to occur when running the spatial GA on a complete graph. In our spatial system instead of mutating the current individual at a location or having it as one of the two parents for crossover, the neighborhood is determined, which includes the node to be filled, and then one or two members (depending on whether crossover is to be done or not) are selected from the entire neighborhood, reproduced and then stored in the next generation's node. When crossover is performed only one of the children are kept (randomly determined). This technique was chosen to match the design decision used in Mitchell et. al. (2006): the only spatial coevolutionary paper to both use and fully describe their spatial crossover technique. If elitism is used, and the current node holds one of the elites, the newly created offspring and the elite member have binary tournament selection applied, implementing local elitism.

As we are attempting to study the effect of elitism we will be changing the number of elites used in order to test its effects. In general we are testing with 0, 1, 2, 3, 5, 10, 50 and 100 (or full) numbers of elites, as stated previously. It is important to remember that we are implementing unique elitism when considering the case where k = 100 (i.e. full local elitism), as this means that there are still some nodes which may be not be elite if there are duplicates at other nodes.

At this point we must point out a minor bias in our elitism selection; one that becomes fairly obvious when we examine the movement of elites in our heat map experiment. When we are selecting our unique elites we sort the nodes by fitness and then remove duplicates. However, the sorting algorithm used is a stable sort, so when it comes across a tie it keeps the nodes in the same order as they occur in the population of which we choose the first as the elite member.

The reproductive spatial structure used in all cases is the standard toroidal grid. This topology was chosen as it was the most common one used in the spatial coevolutionary literature (Hillis, 1990; Wiegand and Sarma, 2004). Mitchell et. al. (2006) also used a toroidal grid, but included the diagonal grid nodes in the neighborhood.

¹ Please note that neither parent need be the elite member itself; see the discussion of how our spatial system is designed in GA Setup.

Coevolution and Spatial Evaluation

For all coevolutionary experiments done in this paper all evaluations are performed using a single member from one population paired with a single member from the other population. In all spatial coevolutionary systems the spatial evaluative structure is a simple one-to-one matching on the reproductive grids. In other words, if a member from one population is located at grid location <5, 7>, it is evaluated along with the member from the other population also located at <5, 7> on its reproductive grid structure.

This method of evaluation has a bleed-over effect on the non-spatial coevolutionary system as implemented. Just as with the standard GA, the non-spatial coevolutionary GA is just the spatial GA run on a complete graph (for both populations). Consequently, our non-spatial coevolutionary system also uses a one-to-one evaluation. This comes about naturally through the implementation, since the exact same coevolutionary system is used, but with the graph being changed from a grid to a complete network. While at first one may think that this is not very much like a standard coevolutionary setup, with some reflection it can be seen that the parents can come from anywhere in the population (since the reproductive structure is completely connected), and so the evaluative pairing is actually random. Thus evaluation has no actual one-to-one pairing of children from the same location within the two populations that, a priori, one might expect with such a setup. The one exception to this comes from the elitism mechanism as elites hold their position independent of where the offspring come from. Therefore, matched elites would still be evaluated against each other and would not shift positions. Thus elites create a stable matching pair through this mechanism, and with many elites, could give the system similar behaviour as the spatial coevolutionary system. We choose to use this elitism style in order to use a cohesive GA for all experiments, spatial and non-spatial; a more general version of elitism will also be explored.

Experimental Design

We will look at the effects of elitism using a cooperative onemax problem and a cooperative one-max matching problem (described below). Although we do have preliminary results for a related competitive function, a fuller exploration will have to wait for future work.

In all cases the problems will be run on the standard GA, the spatial standard GA, the non-spatial coevolutionary GA, and the spatial coevolutionary GA.

Elitism on One-Max

The one-max problem is one of the simplest problems used for evolutionary computational analysis; it is a simple summing of the number of ones in a binary chromosome, with the maximum occurring on a chromosome with all ones. While it is obvious that elitism should help improve the GA's performance on the one-max problem as it is a simple unimodal linearly separable problem, which is extremely amenable to hill climbing, it is not obvious as to what degree it will help. It also allows us to compare the effects of elitism on both a coevolutionary and standard GA using a very easily understood function to better observe the behavior of the systems

without complicating factors. The number of elites investigated are 1, 2, 3, 5, 10 and 100. Finally the one-max problem used has a chromosome length of 60.

The cooperative coevolutionary version of the problem is the same as used for the standard GA, but now the 60-bit solution is split into two halves of 30 bits each. During evaluation the two chromosomes, one from each population, are spliced back together into a single 60 bit one max chromosome which is then scored using same function as the original one-max problem.

Elitism on One-Max Matching

One-max matching is a problem of our own design that combines the one-max problem with a simple matching problem commonly seen in game theory (Chen et. al., 1996). This problem was created in order to produce locals in the unimodal one-max, thus increasing its difficulty in a predictable fashion. Once again, this was done in order to reduce extraneous factors that a more complex problem could present in order to more clearly observe the behaviour of the underlying system. Matching is where one section of a chromosome must match another section in order to gain fitness. Each gene is paired with another gene and if they match the individual gains a fitness point. The result of the matching function is then added to the one-max score that has been calculated across the entire chromosome, though the matching problem's fitness is doubled to get the desired behaviour of matching being equal to one-max (there are twice as many loci to be counted in one-max as there are potential matches to be counted in matching). For our coevolutionary systems, individual members of both populations have "half" the chromosome, each of which is in turn matched within itself (as seen in Figure 1). In other papers in-progress we refer to this function as inner-matching. This is to distinguish it with the basic matching function that matches the chromosome from one population with the chromosome from the other. When basic matching is implemented in the standard GA the first half of

Inner Matching

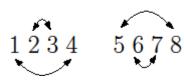
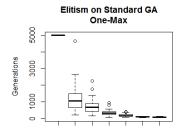
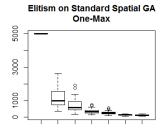


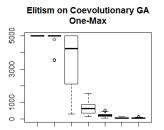
Figure 1: Inner Matching. Genes are matched within the same chromosome.

the chromosome is matched with the second half. As we only implement one-max-innermatching, we shorten its name to simply one-max matching. All one-max-matching experiments use a chromosome length of 32 and elite sizes of 0, 1, 2, 5, 10, 50 and 100.

The one-max matching problem is interesting as it maintains the characteristic of being a simple problem like unimodal one-max while actually being massively multimodal, albeit with very small locals. Consequently, we believe that any observations made on this function have a better chance of generalizing to other fitness functions. The global optimum occurs at the same location as one-max, with all loci set to one, as both one-max and the matching function separately have maximum values at this location. The matching function however has many other equal global maxima; i.e. any chromosome where the two halves of the matching segments are the same. For example using an 8 bit chromosome with two 4 bit matching segments (separated by a space for easy compar-







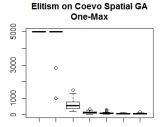


Figure 3: One-max elitism effects on different GA types. Each graph displays the number of generations required to find the solution at each elite size. Notice that the performance improves as we increase the number of elites for all types of GAs.

ison), 1101 1101 will produce a matching value of 4 (and a one-max-matching value of 6+2*4=14). This produces a local maximum, as shown in Figure 2. Changing any of the 0 bits to a 1 will add 1 to the fitness due to one-max, but subtract 2 from the fitness due to loss of a match (which, remember is doubled). If on the other hand you change a 1 bit to a 0 bit, the function will lose both the one-max point and the matching double point. In either case the fitness will drop leading us to conclude that we are at a local maximum. A chromosome length of 32 will result in 2^{32} possible solutions, 2^{30} of which are local maxima, with the global maximum being one of them.

Strictly Non-Local Elitism on One-Max

Here we perform a comparison between (globally selected) local elitism and (globally selected) strict elitism where the elite member is passed into the next generation independent of the local offspring's fitness. As noted previously, when implemented using a complete graph for spatial reproduction, local elitism should have almost identical behavior as regular elitism on a standard GA, but with subtle differences that we believe are insignificant.

To test this hypothesis we implement the standard GA style strict elitism and apply it to all GA types: standard, standard spatial, non-spatial coevolutionary and spatial coevolutionary. We will use elite sizes of 1, 2, 5, 10 and 50. An elite size of 0 is not used because when we have no elitism the style used is irrelevant and so the results would be the same as first experiment (see Figure 3). Also, an elite size of 100 (which we refer to as *full elitism*) is not tested as no chromosome in our population could change as every member would be an elite and would be passed to the next generation unchanged. If our hypothesis is correct, we should see little to no difference

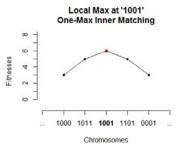


Figure 2: One-max matching's local optimum at '1001'. All chromosomes with a single bit flipped from `1001' are displayed; all cause the fitness to decrease. Note that the maximum fitness for this problem would be 8 at '1111'.

between the two elitism versions on the standard GA. Furthermore, we expect strict elitism to function poorly when compared to local elitism in all systems that have a spatial component. One possible caveat to the above is that the "locked-in" effect that strict elitism might produce in nonspatial coevolutionary systems could lead to unpredictable results.

Heat Maps

We will use the conceptualization tool known as a heat map to track the relative fitnesses of nodes within our spatial GA at various generation points. Viewing the relative fitnesses of these nodes should help us understand the role elitism is playing on the spatial standard and spatial coevolutionary GAs. We will show heat maps of sample runs at elite levels of 0, 2, 5, 10 and 100. We do not bother using this visualization technique on the non-spatial GAs as the layout of fitness value would be completely random in the absence of a reproductive spatial network.

While the heat map looks like a basic grid it is important to remember that it is a representation of a toroidal grid where the left/right and top/bottom sides are connected with one another. Lighter coloured (white) nodes are the nodes with the highest fitness at that generation and darker (red) nodes are the nodes with the worst fitness in that generation. Five heat maps are shown per run: one at the beginning, one near the end, and 3 approximately evenly spaced throughout. Please note that the number of generations between heat maps is variable because of the stochastic nature of the number of generations it takes to find the solution.

Experimental Results

We have found that our experimental data is generally non-parametric using a normal Q-Q plot. Therefore, we will be using non-parametric statistical techniques, such as the box plots to present data and the Wilcoxon Rank Sum test to compare data. The type of box plot we use displays the 'box' around the 25% and 75% quartiles and a bar at the median value. Whiskers are drawn from the box up/down to the most extreme value found within 1.5x of the interquartile range. Dots are then used to display all 'extreme' points that lie beyond the whiskers.

While often you can discern whether a difference is significant or not just by examine the box plots side-by-side, to ensure statistical accuracy, we use a Wilcoxon rank-sum test using a 95% confidence interval along with a Holm-

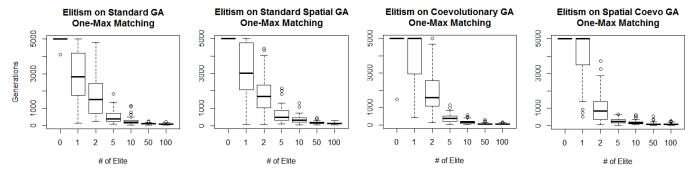


Figure 5: Elitism effect on different GA types solving one-max-matching, a variant of one-max that has local optima. Each graph displays the number of generations required to find the solution at each elite size. Notice that the performance once again improves as we increase the number of elites for all GA types.

Bonferroni post hoc correction. The post hoc correction is applied to ensure that when performing multiple comparisons, all hold simultaneously with a 95 confidence level and not just individually (i.e. controlling the family-wise error rate).

Elitism on One-Max

The results from this experiment can be seen in Figure 3. Each graph displays the number of generations each of the 50 repetitions at each elite size took to find the completed solution. We can clearly see that elitism provides a significant performance increase to all four of our GA types. We also see that we require at least an elite level of 2 for our coevolutionary GA to be able to compete with the standard GA, although we do not yet understand this effect. Once we have enough elite members, elitism obviously helps all of our GA types; however the jump in performance for both the spatial and non-spatial coevolutionary GAs is astounding. Not only does the coevolutionary GA gain an asymmetric amount of performance from elitism, the spatial coevolutionary GA actually overtakes the standard GA in performance by the time we have 10 elites. In addition, the non-spatial coevolutionary GA has roughly the same performance as the standard GA at this elite level. These results can be seen more clearly in Figure 4 with p-values displayed in Table 1. Once we reach full elitism the coevolutionary GA has achieved better performance than either of the standard GA types, while the spatial coevolutionary GA has maintained its performance lead over all of the other GA types.

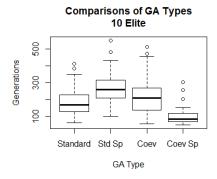


Figure 4: One-max with 10 elites. By 10 elites we see that the coevolutionary spatial GA has surpassed the others in performance (and continues to do so as more elites are added).

GA Type		W	P-Value
Coevo Spatial	Coevolution	259.5	4.39e-11
Coevo Spatial	Std Spatial	96	1.10e-14
Coevo Spatial	Standard	356	2.91e-09
Coevolution	Std Spatial	861	0.0148
Coevolution	Standard	1534.5	0.0502
Std Spatial	Standard	570.5	8.56e-06

Table 1: One-max with 10 elite performance comparisons using a Wilcoxon rank-sum test and a Holm-Bonferroni post hoc correction.

Elitism on One-Max Matching

The one-max matching problem was being tested with the purpose of testing elitism on a problem with local optima. We we're interested in seeing if our elitism mechanism would be hurt by a problem more complex than one max. Viewing the results in Figure 5 we can see that elitism did not hurt the GA systems when solving these problems at all, and in fact full local elitism provided the best performance for this problem on all GA types. We do see, as expected, that the one-max matching problem is a harder problem than standard one-max as we have a shorter chromosome length and yet we are finding worse performance (greater number of generations to find the solution).

We again see that the coevolutionary GAs are receiving a greater gain in performance than the standard GAs for increasing elitism. At an elite value of 2 the coevolutionary spatial GA is already statistically better than any of the other GA types, while the other three are roughly the same. By the time we've reached an elite level of 10, both coevolutionary GA types are better than the standard GA. This is shown in more detail in Figure 6 with p-values displayed in Table 2.

GA Type		W	P-Value
Coevo Spatial	Coevolution	1204	0.754
Coevo Spatial	Std Spatial	1915.5	2.73e-05
Coevo Spatial	Standard	1614.5	0.0443
Coevolution	Std Spatial	588	2.73e-05
Coevolution	Standard	896.5	0.0443
Std Spatial	Standard	1619	0.0443

Table 2: One-max matching with 10 elite performance comparisons using a Wilcoxon rank-sum test and a Holm-Bonferroni post hoc correction.

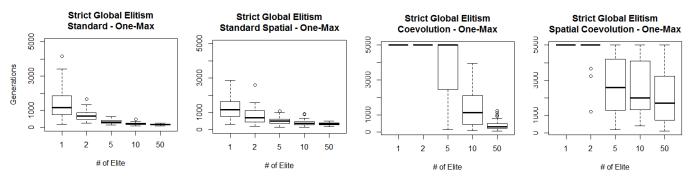


Figure 7: Strictly Non-Local Elitism on different GA types solving one-max. The coevolutionary GAs show an extreme degradation in performance when using this elitism style.

Strictly Non-Local Elitism on One-Max

Strict elitism had a massive effect on the behaviour of the spatial and non-spatial coevolutionary GAs. Using strict elitism the coevolutionary GA's were unable to solve the one-max problem in any of the 50 trials we tried with up to 5 elites. Once we were using 10 strict elites our coevolutionary GA was able to succeed some percentage of the time, but the performance was still up to 10x worse than the standard GA; these results can be seen in Figure 7.

To contrast the massive failure of strict elitism on coevolution, the spatial and non-spatial standard GAs show little to no degradation in performance on the strict elitism style. This is very interesting as we had predicted that our local elites would be an improvement for the spatial GAs, and while there was a minor increase in performance for our standard spatial GA, it was much more so for both of the coevolutionary GAs.

This result somewhat undermines our 'ratcheting' hypothesis, as while this hypothesis may still be true, there is clearly at least something else at play here. We believe that the local elitism may be affecting coevolution due to more of a synergistic effect between the two populations. The elites in the two populations will end up in the same location as both organisms share the same fitness. This allows the population to keep two strong individuals (the elites) 'linked up', and allows them to work together (coevolve) and eventually find the optimum. When we are using strict elitism the linked up indi-

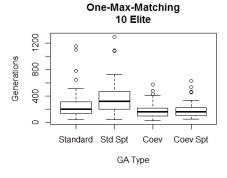


Figure 6: One-max-matching with 10 elites. Again, after 10 elites we see that the coevolutionary GA's performance has improved

viduals are unable to continue evolving as they are 'locked in' to their current set of chromosomes; they are both unable to continue evolving. Instead of the current elites eventually getting more highly fit (or possibly chromosomes nearby), the GA must instead wait until the other individuals in the two populations find their respective optimums at the same reproductive location in the same generation.

Heat Maps

The heat maps are shown in Figure 8 for the coevolutionary and standard GA respectively. We can see a few interesting things from them. The first thing to mention is that we see very little difference when looking at the heat maps of the standard GA versus those of the coevolutionary GA. This implies that the movement of fitness values, and therefore elites, have similar behaviour in both GA types. This is interesting considering the fact that we have shown in the earlier experiments that elites are much more important to the success of the coevolutionary GA than the standard one. This increased success clearly has nothing to do the elites' movement within the population.

Another point of interest is that we see a significant difference in the fitness structure of the GA as we increase the number of elites. We can see that our elites are grouped near each other to form clusters of highly fit nodes. We see evidence of this in all graphs shown, but it is most readily apparent looking at the 3rd and 4th graphs of the 10 elite coevolutionary GA. We can also see this effect looking at the standard GA's graph with 5 elites, though they are focused around the bottom left corner which involves them wrapping around our toroidal structure which can be somewhat hard to visualize on our 2D grid.

The final point to note is the difference between the 0 elite heat maps versus full elite; one might think that these graphs would be the same as we have no elite clustering, however this is not quite true. While the early generations show little to differentiate them, in the later generations the nodes in the full elitism heat maps have a much more average set of data than in the non-elite heat maps. Full elitism appears to not only help the GA find the optimum faster, but also increases the average fitness of the population as well.

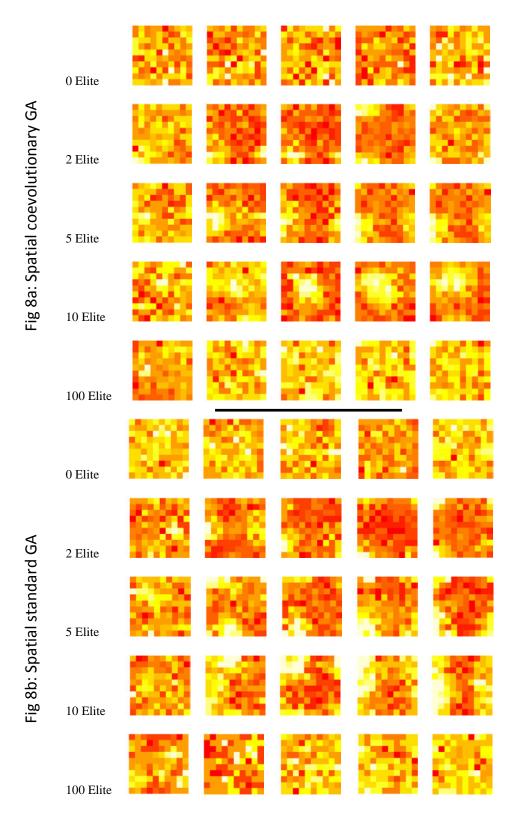


Figure 8: Heat map of fitness values with varying levels of elitism on coevolutionary (Fig 8a) and standard (Fig 8b) GAs. In each of the sub-graphs, the first graph displays the 10^{th} generation (the 1^{st} generation would just be a random seed) and the last graph shows the final generation. The other three graphs are evenly spaced between the first and max generation. The colours in each generation are relative to that generation. The lighter (white) nodes are the most highly fit and the darker (red) nodes are the least fit.

Conclusion

In this paper we analyzed the effects of elitism on a spatial coevolutionary system and found it to be extremely, almost ubiquitously effective. To help in this analysis we developed a globally selected local elitism for the spatial component of the spatial coevolutionary system in order to transition between the strictly global behavior of the elitism mechanism of the standard GA and the local elitism of the spatial GA. This hybrid mechanism demonstrated the effectiveness of local elitism for spatial coevolution over the more strict elitism that ignores the improvements of the offspring. However we were surprised by the reduction of strength of the effect when used in a non-coevolutionary spatial GA.

A-priori it is obvious that elites will always help on unimodal problems such as pure one-max, as the GA cannot be trapped at any local optima. Keeping the best values found so far can only benefit the GA with no downside. After testing the GA on a problem with numerous, albeit small, local optimums (one-max matching) and still seeing a significant increase in performance it is clear that elitism will still be helpful on all but the most deceptive of problems. Our version of local elitism allows evolution to occur once nodes across populations becomes 'locked in' through the elitism and continually 'ratchets up' their combined fitness in a synergistic manner. While this could theoretically allow the local neighborhood within the spatial coevolutionary GA to be trapped at a "local optima", we hypothesize that the global selection component of the mechanism allows these "locked in" pairs to be separated by the reproductive grid structure and thus allowing for gene-flow from nearby neighbors to allow the system to escape. Finally we observed that it required two elites in both of the coevolutionary populations for the extreme beneficial effects to be realized. Why one elite member per population was insufficient to see any measurable benefit cannot yet be explained.

Acknowledgements

We wish to acknowledge the support and funding of the Natural Sciences and Engineering Research Council of Canada for this research.

References

- Chen, S., Duffy, J. and Yeh, C. (1996). Genetic programming in the coordination game with a chaotic best-response function. *Proceedings of the 5th Annual Conference on Evolutionary Programming*, 277-286.
- De Jong, K. (1975). *Analysis of the behavior of a class of genetic adaptive systems.* Ph.D. thesis, Computer and Communications Science Department, University of Michigan.
- De Jong, K., and Sarma, J. (1995). On Decentralizing Selection Algorithms. *ICGA*. Vol. 95.
- Hillis, W. D. (1990). Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*: 42.1: 228-234.
- Mitchell, M., Thomure, M. and Williams, N. (2006). The role of space in the success of coevolutionary learning. *Artificial Life X*, 118-124.Sarma, J. and De Jong, K. (1996). An analysis of the effects of

- neighborhood size and shape on local selection algorithms. *Parallel Problem Solving From Nature IV*, 236-244.
- Pagie, L. and Hogeweg, P. (1997). Evolutionary consequences of coevolving targets. Evolutionary Computation 5(4):401-418.
- Potter, M. A. and De Jong, K. (1994). A cooperative coevolutionary approach to function optimization. *Parallel Problem Solving From Nature* III, 249-257.
- Wiegand, R. P. and Sarma, J. (2004). Spatial embedding and loss of gradient in cooperative coevolutionary algorithms. *Parallel Problem Solving From Nature VIII*, 912-921.
- Williams, N. and Mitchell, M. (2005). Investigating the success of spatial coevolution. *Proceedings of the 7th annual conference on Genetic and Evolutionary Computation*, 523-530

Pleasure, Persistence and Opportunism in Action Selection

Matthew Lewis¹, Antoine Hiolle¹ and Lola Cañamero¹

¹Embodied Emotion, Cognition and (Inter-)Action Lab School of Computer Science, University of Hertfordshire College Lane, Hatfield, Herts, AL10 9AB, U.K. matt-1@semiprime.com, L.Canamero@herts.ac.uk

Abstract

An autonomous robot must show appropriate levels of persistence and opportunism to survive. We address this problem by using a mechanism akin to pleasure that modulates exteroception as a function of need satisfaction, rather than based on internal deficits and external threats as in previous work. The different context in which the modulating hormone is released has important consequences on persistence and opportunism.

Introduction

Persistence and opportunism are two key features of action selection architectures (Tyrrell, 1993; Maes, 1995). For an autonomous robot that has to satisfy multiple conflicting survival-related needs, it is crucial not only to choose behaviors that do so in a timely fashion, but also to persist in their execution for long enough to guarantee sufficient satisfaction. Persistence is important to avoid what is known as the "dithering" problem, which occurs when a robot keeps switching between trying to satisfy two needs without satisfying either of them enough to guarantee survival. Another key feature is opportunism: to consume a resource that might not be needed at present but is available now and might not be available later. The degree to which a robot should show persistence and opportunism depends on multiple factors; we could generally say that persistence leads to a more "conservative" action selection behavior and opportunism to a more "risky" one. In previous work (Avila-García and Cañamero, 2004), we showed that persistence and opportunism can also become negative when done in excess, and proposed a mechanism inspired by emotions in natural systems and based on hormonal modulation of the perception of external stimuli (the resources), to address these problems. In that work, the hormone modulating perception was released as a function of internal deficits and the presence of threats in the environment, i.e., it was released signaling that things were not functioning well. Building on that work, here we present a related motivated action selection architecture that uses a mechanism akin to pleasure (also modeled using artificial hormones) to modulate the perception of the resources. Following the principle that pleasure signals well functioning (Panksepp, 1998), the "pleasure hormone" is released as a function of need satisfaction. The very different context in which the hormone that modulates perception is released creates different behavioral dynamics and has important consequences regarding how the architecture addresses persistence and opportunism.

Robot Architecture

Following a definition of autonomy as self-regulation, we use a homeostatically-controlled motivated architecture having to solve a two-resource action selection problem, in the same vein as Avila-García and Cañamero (2004). Our robot (an Aldebaran humanoid Nao) has thus two homeostatic internal variables - energy and moisture deficits, ranging between 0 and 100, where 100 is the fatal limit - that, in conjunction with the stimuli in the environment, motivate it to select the appropriate behavior to execute. Perception of the external environment is done using a camera to detect the resources – colored balls used as "food" and "drink" – and sonar and contact sensors to move around safely. Each resource, when consumed by the robot, reduces the corresponding simulated deficit. Fig. 1 shows the experimental setup and Fig. 2 the architecture. Two motivations thirst and hunger - each corresponding to a deficit, combine the perception of internal deficits and external stimuli to guide behavior selection. As in Avila-García and Cañamero (2004) we calculate each motivation level using the formula

$$motivation_i = deficit_i + (deficit_i \times \alpha \times cue_i)$$
 (1)

where cue_i is the magnitude of the stimulus – here perceived size of a resource related to the deficit – and α modulates the perception of resources. In this case, we have used a "pleasure" hormone that we hypothesized can allow persistence while reducing pathological opportunistic behavior. Motivations influence which behaviors are selected for execution (in 8Hz-tick cycles) by passing them their activation level. Each behavior – a perception-action loop – has a self-controlled activation level that depends on sensory inputs that are relevant for it, and activation threshold, and becomes potentially executable when its activation level is above the threshold. For this study we have implemented two top-level behavioral systems – "behaviors" for short – each linked

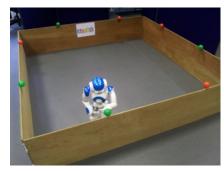


Figure 1: Experimental setup. The green balls around the arena represent drink resources, and the red food resources.

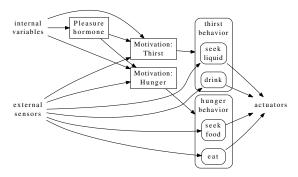
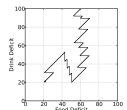


Figure 2: Simplified robot architecture

to a motivation. These *thirst* and *hunger* top-level behaviors are implemented by embedding simpler behaviors, both consummatory (*eat* food and *drink*, which can be executed only if the corresponding stimulus is present) and appetitive (goal-seeking, including wandering randomly, scanning the environment, tracking a detected object, and walking in the direction that the head is turned), to seek out the relevant environmental resources, and consume them if found. The selected top-level behavior in turn acts as a behavior selection mechanism selecting from amongst its sub-behaviors. While the top-level behaviors are selected on a winner-takeall basis by motivations, multiple lower-level behaviors can be run simultaneously, provided they use disjoint sets of actuators.

Experiments and Results

To investigate our hypothesis, we modulated exteroception using a "pleasure" hormone released when either of our deficits decreases. Subsequently, the level of the hormone decays back to its basal level. Intuitively, the hormone indicates that recent behaviors were beneficial. We tested two conditions: a modulated α in equation 1 proportional to the level of the hormone, and a control condition using a constant α . Results from two runs are shown in Fig. 3. In the case where α is *constant* (left), the robot displays adaptive persistence in its drinking behavior soon after starting (Food deficit in the 40s). However, not long after this, with the Food deficit around 60 it eats, and keeps eating opportunistically while the Drink Deficit continues to rise. The robot



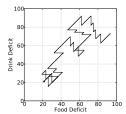


Figure 3: Results of two runs showing the evolution of deficits over time. Left: constant $\alpha=0.5$ (control). Right: α modulated by pleasure hormone.

dies of thirst in less than four minutes. In the case where α is modulated by the pleasure hormone (right), the robot shows persistence (particularly to the top right of the graph), but avoids getting stuck performing behaviors opportunistically. Although it has high Food and Drink deficits, it is still alive after more than five minutes.

Discussion

Our preliminary results are consistent with the hypothesis that in some environments (excessive) opportunistic behavior can be maladaptive, and that using pleasure to modulate the α parameter in our motivation formula can offer a solution to this problem, while still allowing persistence. Using pleasure in this way means that opportunistic behavior can be reduced overall, but cues that are present during the release of the hormone will have their significance magnified until the hormone has decayed back to its basal level. Since it is likely (but not guaranteed) that these cues influenced the behaviors that caused the drop in the deficits, and the subsequent release of the hormone, pleasure would be adaptively communicating the instruction "keep acting in response to any cues that are present". This is consistent with the role of positive affect in fostering openness towards the world. This also opens up the possibility of using other sources of pleasure to make behaviors persist, i.e. to externally influence its actions. When interacting with a human, social pleasure could be used as a tool to reinforce behaviors in the robot.

Acknowledgments

This work was supported by EC grant FP7-ICT-248116 (ALIZ-E). We are grateful to the other members of the Embodied Emotion, Cognition and (Inter-)Action Lab for discussions on this research. The opinions expressed are solely the authors'.

References

Avila-García, O. and Cañamero, L. (2004). Using hormonal feedback to modulate action selection in a competitive scenario. In *Proc. SAB'04*, pages 243–252. MIT Press.

Maes, P. (1995). Modeling adaptive autonomous agents. *Artificial Life: An Overview*, pages 135–162.

Panksepp, J. (1998). Affective Neuroscience. OUP.

Tyrrell, T. (1993). Computational Mechanisms for Action Selection. PhD thesis, University of Edinburgh.

A continuous-time binary consensus protocol with hysteretic units applied to the density classification problem

Marco A. Montes de Oca and Louis F. Rossi

Dept. of Mathematical Sciences, University of Delaware, Newark, DE 19716 marco@montes-de-oca.com, rossi@math.udel.edu

Abstract

In this paper, we present a continuous-time binary consensus protocol whereby entities connected via a directed ring topology solve the one-dimensional density classification problem. In our model, the participating entities behave as non-ideal relays, that is, they have memory of the trajectory of an internal state variable, which gives them hysteretic properties. We show that this feature is necessary for the system to reach consensus on the state shared by the initial majority. The connections between this protocol and collective decision-making mechanisms in swarm intelligence systems are also discussed.

Introduction

The density classification problem (also known as the majority problem), consists in classifying finite linear binary strings according to whether they have a majority of 0's or 1's. This problem has been the subject of numerous studies in the cellular automata literature (see, e.g., Mitchell et al. (1994); Land and Belew (1995); Fukś (1997, 2002); Alonso-Sanz and Bull (2009); Fatès (2013)) because it has a simple formulation and illustrates very well the idea of "emergent computation" since the cells interact only locally and do not have access to the global structure they are trying to classify. In a cellular automata setting, the density classification problem translates into finding evolution rules for cells that make the automata be all equal to 0 if initially there were more than half of the cells in a 0 state, or 1 if initially there were more than half of the cells in a 1 state.

The idea of emergent computation is also present in another computational paradigm called swarm intelligence (Bonabeau et al., 1999; Dorigo and Birattari, 2007). In this paradigm, relatively simple agents locally interact with one another and with their environment to produce self-organized spatio-temporal patterns that represent solutions to problems that no individual agent could solve on its own (e.g., finding shortest paths (Goss et al., 1989), sorting (Deneubourg et al., 1990), or constructing nests (Grassé, 1959)).

As we explain in the next section, the density classification problem is relevant in swarm intelligence because some of the methods that solve it may be used as collective decision-making mechanisms for swarms. In this paper, we further explore the connection between cellular automata and swarm intelligence by presenting and analyzing a consensus protocol¹ on networks of agents derived from previous work on collective decision-making in swarms (Montes de Oca et al., 2012).

Our consensus protocol may be seen as encoding evolution rules for continuous-time cellular automata with memory. It may also be thought of as a model of social influence in a group whose members observe the actions performed by other individuals, increasing, as a result, their tendency to perform the observed actions. We are interested in this type of mechanisms because we want to eventually endow swarms of robots or agents with collective-decision mechanisms that are robust, flexible and effective in real environments. Our approach is backed by recent studies on social learning (Rendell et al., 2010), that support the idea that learning from the observation of others' actions is a mechanism whereby individuals indirectly probe the environment. In a swarm, which is typically composed of many individuals, using the behavior of others as a guide provides each individual with potentially many indirect channels for obtaining information about their environment, and thereby increasing the amount of information they use to make deci-

The key finding presented in this paper is that if agents influence each other as if arranged in a unidirectional ring, and each agent integrates over time information coming to it from its neighbor using a mechanism akin to exponential smoothing (Gardner Jr., 2006), then symmetric blocks of 0's or 1's propagate through the network indefinitely. Moreover, we provide evidence that when the symmetry of these blocks is broken (that is, there is a majority of 0's or 1's), then the information wave propagates for a finite amount of time and eventually dies out, which translates into the population of agents reaching a consensus. Finally, the state on which the

¹We use the term *protocol* to comply with literature tradition in communication networks and where interaction rules are called protocols. See, for example, (Mesbahi and Egerstedt, 2010).

population reaches a consensus corresponds to that of the initial majority. In other words, we provide evidence that the proposed continuous-time consensus protocol with hysteretic units solves the density classification problem. Ongoing work is aimed at analytically determining how much time is needed for the system to converge.

Collective Decision-Making in Swarms

In (Montes de Oca et al., 2012), we proposed a social influence model whose dynamics can be used as a collective decision-making mechanism for swarms of robots that need to collectively choose the most efficient of two alternative actions (henceforth referred to as *Decision-Making Model or DM model*). In the DM model, each of a set of n agents can be in one of two states (represented with a binary variable $X_i \in \{0,1\}$, with $i=1,2,\ldots,n$). In applications of the DM model, an agent's state can represent, for example, a robot's preferred action or current belief of the state of an environmental variable.

The DM model is a discrete-time model where at each time step t of the system's evolution, an agent i might be able to observe the state of another random agent $j \neq i$. When agent i observes the state of another agent j, the observing agent i updates an internal real-valued variable S_i , which we call tendency, as follows:

$$S_i^{t+1} = (1 - \alpha)S_i^t + \alpha X_i^t,$$
 (1)

where $0 \le \alpha \le 1$ determines the relative weight given to the agent's latest observation (X_j^t) and the agent's accumulated experience (S_i^t) .

After updating its tendency, an agent updates its state as follows:

$$X_{i}^{t+1} = \begin{cases} 1, & \text{if } S_{i}^{t+1} \geq \lambda \\ 0, & \text{if } S_{i}^{t+1} \leq \mu \\ X_{i}^{t}, & \text{if } \mu < S_{i}^{t+1} < \lambda, \end{cases}$$
 (2)

where $\mu+\lambda=1$ (the reason for this constraint will become apparent later). Eq. 2 implements a sort of dynamic memory that allows the agent to integrate its observations over time. By properly choosing values for the parameters α,μ,λ , and the initial conditions X_i^0 and S_i^0 , one can control the imitation behavior of agent i. While in principle, each agent may have different values for its parameters, in the DM model, α , λ and μ , are constant and common to all agents. An example of the behavior of an individual agent in the DM model is shown in Figure 1.

In the DM model the population is reshuffled randomly so that each individual observes a different agent at each time step. Additionally, one single agent may potentially influence more than one other agent in the group. The DM model's collective dynamics make the population reach a consensus on the state that at time step 0 is shared by most (that is, the majority) of the population. In (Montes de Oca

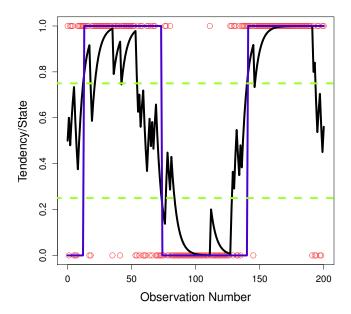


Figure 1: Single agent behavior in the DM model. Starting with an initialization of $S^0=0.5$ and $X^0=0$, an agent observes a stream of state values plotted as dots with values 0 or 1. The black line shows the evolution of the agent's tendency and the blue line shows the evolution of the agent's state. In this simulation, $\alpha=0.2$, $\lambda=0.75$ and $\mu=0.25$ (shown as dotted lines).

et al., 2012), and (Montes de Oca et al., 2011), we show how this behavior can be used for optimal collective decision-making in robot swarms.

The DM model may be seen as a method to solve the density classification problem if its definition is relaxed. In particular, if the agents (cells) are allowed to be reshuffled, then the DM may solve it. In this paper, we explore the question of whether it is possible to solve the original density classification problem with a variation of the DM model that does not require reshuffling. In the following sections, we present such a variation as well as theoretical and experimental results that make us believe that the question can be answered positively.

Continuous-Time Consensus Protocol with Hysteretic Units

The protocol that we propose in this paper, henceforth referred to as *Consensus with Hysteresis or CH*, is in its basic form the continuous-time equivalent of Eq. 1. However, in the CH protocol, the communication topology of the population does not change over time and each agent influences exactly one other agent. In the remainder of this paper, we assume that individuals are arranged in a directed ring topology with an agent i influenced by agent j, where j = i+1 or j = i-1 (agents "look" to their right or left, respectively).

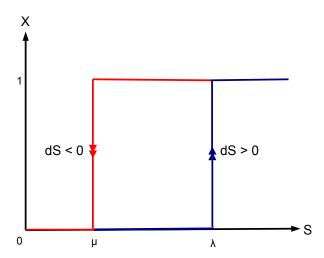


Figure 2: Hysteresis loop. Since hysteresis is nonreversible, we always assume that dt>0. If dS>0 (the input increases), the output evolves according to $\frac{dX}{dt}=g_1(S,X)$ (blue line). If dS<0 (the input decreases), the output evolves according to $\frac{dX}{dt}=g_2(S,X)$ (red line).

In the CH protocol, the tendency-update equation for an agent i influenced by agent j ($i \neq j$) is:

$$\frac{dS_i}{dt} = \alpha(X_j - S_i), \qquad (3)$$

where S_i is now a continuous-time function and X_j is *not* a function but a nonlinear operator that defines a nonideal relay on S_j , that is, $X_j = R_{\mu,\lambda}(S_j)$, where $R_{\mu,\lambda}$ is a hysteresis operator with thresholds μ and λ (see Figures 2 and 3). In this work, we use the Duhem model of hysteresis (Visintin, 2006) to model this operator. The Duhem model defines X_j as the solution of the initial value problem:

$$\begin{cases}
\frac{dX_j}{dt} = g_1(S_j, X_j) \left(\frac{dS_j}{dt}\right)^+ - g_2(S_j, X_j) \left(\frac{dS_j}{dt}\right)^-, \\
X_j(0) = X_j^0,
\end{cases}$$
(4)

where $x^+ = (|x|+x)/2$ and $x^- = (|x|-x)/2$ for any $x \in \mathbb{R}$. Additionally, g_1 and g_2 are given nonnegative functions that represent the paths of evolution of the pair (S, X) for increasing and decreasing S, respectively (see Figure 2).

Rewriting Eq. 3 into standard form, we have:

$$\frac{dS_i}{dt} + \alpha S_i = \alpha X_j \,,$$

whose solution may be found by using the integrating factor $e^{\int \alpha dt}=e^{\alpha t}$. After multiplying and rearranging, we obtain

$$\frac{d}{dt}\left(e^{\alpha t}S_i\right) = e^{\alpha t}\alpha X_j$$

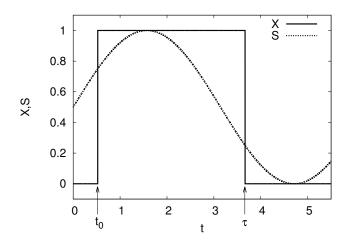


Figure 3: Hysteresis output of a nonideal relay operator $X = R_{\mu,\lambda}(S)$ with $\mu = 0.25$ and $\lambda = 0.75$.

Integrating both sides from t_0 to t:

$$\int_{t_0}^t \frac{d}{dw} \left(e^{\alpha w} S_i \right) dw = \int_{t_0}^t e^{\alpha w} \alpha X_j dw.$$

The right-hand side of this equation may be integrated by parts with $u'=\alpha e^{\alpha w}$, and $v=X_j$, which means $u=e^{\alpha w}$ and $v'=\frac{dX_j}{dw}$. After some substitutions and simplifications, we obtain:

$$S_{i}(t) = X_{j}(t) + e^{\alpha(t_{0} - t)} (S_{i}(t_{0}) - X_{j}(t_{0})) - e^{-\alpha t} \int_{t_{0}}^{t} e^{\alpha w} \frac{dX_{j}}{dw} dw.$$
(5)

To completely integrate Eq. 5, we must consider two cases: case 1 for $dS_j>0$, and case 2 for $dS_j<0$. For case 1, Eq. 5 becomes

$$S_{i}(t) = X_{j}(t) + e^{\alpha(t_{0} - t)} (S_{i}(t_{0}) - X_{j}(t_{0})) - e^{-\alpha t} \int_{t_{0}}^{t} e^{\alpha w} g_{1}(S_{j}, X_{j}) dw.$$

$$(6)$$

Similarly, for case 2, Eq. 5 becomes

$$S_{i}(t) = X_{j}(t) + e^{\alpha(t_{0} - t)} (S_{i}(t_{0}) - X_{j}(t_{0})) - e^{-\alpha t} \int_{t_{0}}^{t} e^{\alpha w} g_{2}(S_{j}, X_{j}) dw.$$

$$(7)$$

In our case, since the hysteresis loop may be seen as being composed of step functions, $\frac{dX_j}{dt} = g_1(S_j, X_j) = g_2(S_j, X_j) = 0$ in the interval $t \in (t_0, \tau)$, where t_0 is the time at which X last switched from 0 to 1 or vice versa, and τ is the time at which X next switches from 0 to 1 or vice versa (see Figure 3).

Table 1: Possible cases for the transition of states of agent i influenced by agent j.

Con	dition	Result		
$\overline{X_i}$	X_j	dS_i	State transition	
0	0	≤ 0	No	
0	1	> 0	Yes, if $S_i < \lambda$	
1	0	< 0	Yes, if $S_i > \mu$	
1	1	≥ 0	No	

Therefore, in the time interval (t_0, τ) , the tendency of any agent i is given by

$$S_i(t) = X_j(t) + e^{\alpha(t_0 - t)} (S_i(t_0) - X_j(t_0)).$$
 (8)

Eq. 8 can be further simplified by recalling that in the interval (t_0, τ) , X_i does not change, therefore

$$S_i(t) = (1 - e^{\alpha(t_0 - t)})X_i(t_0) + e^{\alpha(t_0 - t)}S_i(t_0).$$
 (9)

This last equation clearly shows that in intervals beyond (t_0,τ) , $S_i(t)$ is actually a piecewise function that depends on how $X_j(t)$ evolves over time. Therefore, the CH protocol has a punctuated evolution, which allows us to analyze the dynamics of a system of agents governed by Eq. 9 by breaking time into intervals defined by a sequence of τ 's. For example, we start with an interval $(t_0=0,\tau=\tau_1)$, then continue with $(t_0=\tau_1,\tau=\tau_2)$, then with $(t_0=\tau_2,\tau=\tau_3)$ and so forth, where $\tau_1,\tau_2,\tau_3,\ldots,\tau_k$ are the times at which some agent in the population switches from 0 to 1 or vice versa. Such an analysis is presented in the next section.

Consensus with Hysteretic Units: Analysis and Simulations

We focus first on the calculation of the sequence of τ 's at which the system's states change as described above. In order to do this, we first notice that only a subset of all the possible initial conditions can lead to state transitions (see Table 1).

A state transition is possible only if the agents involved in an interaction have opposite states (this is an important observation because *it implies that consensus is an absorbing state*). In such cases, it is possible to calculate the time between state transitions τ .

In the second case of Table 1, $X_i(t_0)=0$, $X_j(t_0)=1$, and $S_i(t_0)<\lambda$, we have

$$S_i(\tau) = X_j(\tau) + e^{\alpha(t_0 - \tau)} \left(S_i(t_0) - X_j(t_0) \right) = \lambda.$$

Solving this equation for τ , we obtain

$$\tau = t_0 + \frac{1}{\alpha} \ln \left(\frac{X_j(t_0) - S_i(t_0)}{X_j(\tau) - \lambda} \right)$$

Since we are assuming that in the interval $(t_0, \tau]$, X_j does not change, then $X_j(\tau) = X_j(t_0) = 1$. Thus, we have

$$\tau = t_0 + \frac{1}{\alpha} \ln \left(\frac{1 - S_i(t_0)}{1 - \lambda} \right). \tag{10}$$

Similarly, in the third case of Table 1, $X_i(t_0)=1$, $X_j(t_0)=0$, and $S_i(t_0)>\mu$, we have

$$S_i(\tau) = X_j(\tau) + e^{\alpha(t_0 - \tau)} \left(S_i(t_0) - X_j(t_0) \right) = \mu.$$

Solving this equation for τ , we obtain

$$\tau = t_0 + \frac{1}{\alpha} \ln \left(\frac{X_j(t_0) - S_i(t_0)}{X_j(\tau) - \mu} \right).$$

In this case, if $X_j(\tau) = X_j(t_0) = 0$, then

$$\tau = t_0 + \frac{1}{\alpha} \ln \left(\frac{S_i(t_0)}{\mu} \right) . \tag{11}$$

From Eqs. 10 and 11, it is clear that the time to transition from 0 to 1 and from 1 to 0 is equal only when $S_i(t_0)=\frac{1}{2}$ and $\lambda+\mu=1$ (this is the origin of the restriction in Eq. 2). This result is important in connection with the following observation.

That there is no state transition when $X_i = X_j$, does not mean that the tendency S_i does not change. In fact, if $X_i = X_j = 0$ and $S_i(t_0) > 0$, then $dS_i < 0$ and S_i decreases. Similarly, if $X_i = X_j = 1$ and $S_i(t_0) < 1$, then $dS_i > 0$ and S_i increases. This fact makes contiguous blocks of 0's or 1's particularly interesting because they make agents' tendencies go over or below the hysteresis thresholds and delay future state transitions (possibly asymmetrically). Let us illustrate with two simple examples the situations that we can encounter as a consequence of having symmetric or asymmetric blocks of 0's or 1's.

Example 1: 0011 (symmetric blocks of 0's and 1's)

Imagine we initialize our units with the pattern 0011 (see Figure 4), that is, $X_1(0)=0$, $X_2(0)=0$, $X_3(0)=1$, and $X_4(0)=1$. Also, assume that $S_i(0)=1/2$ for $i\in\{1,2,3,4\}$. Table 2 shows the evolution of the system for the first 4 transitions.

This example shows that while the initial conditions for the agents' tendencies change over time, the symmetry of the system is kept and therefore, the state of the system is stable (more about this later). The duration of the inter-state transition intervals $\Delta \tau = \tau_{k+1} - \tau_k$ for some k is determined by the units that are at the interface between blocks of 0's and 1's. It is interesting to see the evolution of the duration of these intervals over time, especially for relatively large populations. In Figure 5, we show the duration of the interstate transition intervals for a system with 10 units divided into 2 symmetric blocks of 0's and 1's.

Table 2: Punctuated evolution of a system of 4 units initialized with $S_i(0) = 1/2$ for $i \in \{1, 2, 3, 4\}$ and $X_1(0) = 0$, $X_2(0) = 0$, $X_3(0) = 1$, and $X_4(0) = 1$.

\overline{t}	X_1	S_1	X_2	S_2	X_3	S_3	X_4	S_4
0	0	$\frac{1}{2}$	0	$\frac{1}{2}$	1	$\frac{1}{2}$	1	$\frac{1}{2}$
$\tau_1 = \frac{1}{\alpha} \ln(\frac{1}{2\mu})$	1	λ	0	μ	0	μ	1	λ
$\tau_2 = \tau_1 + \frac{1}{\alpha} \ln(\frac{\lambda}{\mu})$	1	$1 - \frac{\mu^2}{\lambda}$	1	λ	0	$\frac{\mu^2}{\lambda}$	0	μ
$\tau_3 = \tau_2 + \frac{1}{\alpha} \ln \left(\frac{1 - \frac{\mu^2}{\lambda}}{\mu} \right)$	0	μ	1	$1 - \frac{\mu^2}{1 - \frac{\mu^2}{\lambda}}$	1	λ	0	$\frac{\mu^2}{1 - \frac{\mu^2}{\lambda}}$
$\tau_4 = \tau_3 + \frac{1}{\alpha} \ln \left(\frac{1 - \frac{\mu^2}{1 - \frac{\mu^2}{\lambda}}}{\mu} \right)$	0	$\frac{\mu^2}{1 - \frac{\mu^2}{1 - \frac{\mu^2}{\lambda}}}$	0	μ	1	$1 - \frac{\mu^2}{1 - \frac{\mu^2}{1 - \frac{\mu^2}{\lambda}}}$	1	λ

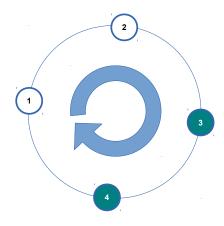


Figure 4: Example initialization of the CH protocol with four units. The direction of information flow is depicted with the big circular arrow.

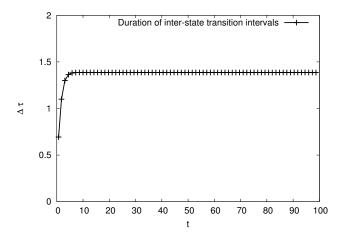


Figure 5: Duration of the inter-state transition intervals over time for a system of 10 units initialized with $S_i(0)=1/2$ for $i=1,\ldots,10$ and $X_i(0)=0$, for $1\leq i\leq 5$, and $X_j(0)=1$ for $6\leq j\leq 10$. Additionally, $\alpha=1$, $\mu=0.25$, $\lambda=0.75$

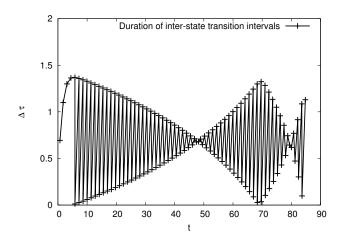


Figure 6: Duration of the inter-state transition intervals over time for a system of 10 units initialized with $S_i(0) = 1/2$ for $i = 1, \ldots, 10$ and $X_i(0) = 0$, for $1 \le i \le 3$, and $X_j(0) = 1$ for $4 \le j \le 10$. Additionally, $\alpha = 1$, $\mu = 0.25$, $\lambda = 0.75$

Example 2: 0111 (asymmetric blocks of 0's and 1's)

Now consider an initial pattern 0111, that is, $X_1(0)=0$, $X_2(0)=1$, $X_3(0)=1$, and $X_4(0)=1$. Also, assume that $S_i(0)=1/2$ for $i\in\{1,2,3,4\}$. Table 3 shows the evolution of the system for the first 4 transitions.

This example shows that when there are blocks of 0's or 1's of asymmetric length, there are eventually asymmetric initial conditions on the agents' tendencies that makes one agent (in our example, agent 3) switch state before another agent (in our case, agent 4). Through this process, the state of blocks of longer length propagate through the network, which eventually leads to consensus on the initial majority.

The duration of the inter-state transition intervals in the case of asymmetric blocks of 0's or 1's is no longer simple as in the case of symmetric blocks as shown in Figure 6. This figure shows how state updates can become out of sync in asymmetric settings.

It should be clear that in our proposed protocol, informa-

13(0) = 1, and $214(0) = 1$.					
\overline{t}	X_1 S_1	X_2 S_2	X_3 S_3	X_4	S_4
0	$0 \frac{1}{2}$	$1 \frac{1}{2}$	$1 \frac{1}{2}$	1	$\frac{1}{2}$
$\tau_1 = \frac{1}{\alpha} \ln(\frac{1}{2\mu})$	1λ	0μ	1 λ	1	λ
$ au_2 = au_1 + rac{1}{lpha} \ln(rac{\lambda}{\mu})$	$1 1 - \frac{\mu^2}{\lambda}$	1λ	0μ	1	$1 - \frac{\mu^2}{\lambda}$
$\tau_3 = \tau_2 + \min\left\{\frac{1}{\alpha}\ln\left(\frac{\lambda}{\mu}\right), \frac{1}{\alpha}\ln\left(\frac{1-\frac{\mu^2}{\lambda}}{\mu}\right)\right\} = \tau_2 + \frac{1}{\alpha}\ln\left(\frac{\lambda}{\mu}\right)$	$1 1 - \frac{\mu^3}{\lambda^2}$	$1 1 - \frac{\mu^2}{\lambda}$	1 λ	1	$\frac{\mu}{\lambda} \left(1 - \frac{\mu^2}{\lambda} \right)$

Table 3: Punctuated evolution of a system of 4 units initialized with $S_i(0) = 1/2$ for $i \in \{1, 2, 3, 4\}$ and $X_1(0) = 0$, $X_2(0) = 1$, $X_3(0) = 1$, and $X_4(0) = 1$.

tion propagates from one unit to another as waves that either keep their period, or that changes over time depending on whether there are equal numbers of 0's or 1's in the system.

 ∞

Let us now address the question of whether it is possible for a localized region of consensus to exist or not. That is, is it possible to sustain a configuration where $X_i=0$ for all i except for a finite set of contiguous agents? We assume for the analysis that follows that each agent is influenced by the next consecutive agent: j=i+1 for all j.

To simplify the problem, we will search for propagating solutions where initially $X_i=1$ for $i\in[1,m]$. We seek solutions for which the set propagates synchronously to the left, i.e., X_0 transitions to 1 as X_m transitions to 0, and determine what boundary conditions S_0 at i=0 and S_m at i=m are required for a sustained synchronized translation to the left as shown in Figure 7.

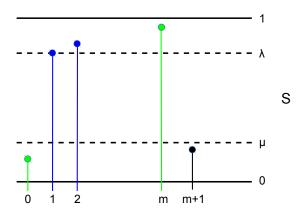


Figure 7: A schematic diagram of a localized region of consensus. The values of S for the translating region of 1's are shown in blue. The boundary agents are indicated in green.

At the leading edge of the region of consensus, we can determine the time Δt_l required for X_0 to transition to 1 after agent 1 transitions to a 1 $(S_1 = \lambda)$. In fact, we immediately see that for a translating region of consensus,

$$S_k = 1 - \mu e^{-\alpha(k-1)\Delta t_l}, \ k \in [1, m].$$
 (12)

Also, we see the transition must satisfy the relation,

1

$$\lambda = 1 - (1 - S_0)e^{-\alpha \Delta t_l} \tag{13}$$

1

because $X_1 = 1$. We note that S_0 is an imposed boundary condition. Therefore,

$$\Delta t_l = \frac{1}{\alpha} \ln \left(\frac{1 - S_0}{\mu} \right). \tag{14}$$

Meeting the requirement that $\Delta t_l > 0$ is nothing more than requiring $S_0 < \lambda$ which is self-consistent with a transition from $X_0 = 0$ to $X_0 = 1$.

At the trailing edge of the region of consensus, we perform the same calculation. In this case, we can calculate the time Δt_t required for X_m to transition to 0 from 1. Similarly, this event must satisfy the relation

$$\mu = S_m e^{-\alpha \Delta t_t},\tag{15}$$

so that

$$\Delta t_t = \frac{1}{\alpha} \ln \left(\frac{S_m}{\mu} \right). \tag{16}$$

To enforce a synchronous transition, we require $\Delta t_l = \Delta t_t = \Delta t$, so that the boundary conditions satisfy $S_0 = 1 - S_m$:

$$S_0 = \mu e^{-\alpha(m-1)\Delta t}. (17)$$

Substituting (14) into (17) and simplifying, we can determine S_0 implicitly:

$$\frac{S_0}{(1-S_0)^{m-1}} = \mu. (18)$$

The solution described by (12), (14) with (18) is artificial in the sense that we cannot specify values of S for the leading agents and regardless they are evolving in time. However, if we consider a cyclic network geometry of length 2m, we can construct a translating region using this solution for elements 1 through m and its image with $X_k=0$ and $S_k=1-S_{k-m}$ for $k\in[m+1,2m]$. In fact, one can construct arrays of such solutions on networks of length 2m, 4m and so forth. The structure of these solutions and lack of free parameters suggest that localized consensus is only possible when there are equal numbers of 0's and 1's.

As we noted earlier, slight perturbations of these solutions lose their synchronization, so the leading and trailing edges will not transition at the same time, and many open questions remain about the stability and evolution of these configurations.

Conclusions

The consensus protocol presented in this paper is derived from previous work of ours on collective decision-making in swarms (Montes de Oca et al., 2012). While the protocol may be seen as a simplification of the full model presented in that work, the protocol exhibits a surprisingly rich dynamical behavior, especially when the number of units with initial state equal to 0 and 1 are not the same. In this case, the protocol effectively solves the density classification problem. We also showed that when the number of initial 0's and 1's is the same, distributed either as large contiguous blocks of 0's or 1's or as interspersed blocks of equal length, the information waves generated by the protocol travel indefinitely across the network of units that form the system.

We are currently working on analytical solutions that could help us answer questions regarding the system's ability to classify strings of any length, and with arbitrary distributions of 0's and 1's within these strings. Also of interest is the determination of the time needed for convergence.

References

- Alonso-Sanz, R. and Bull, L. (2009). A very effective density classifier two-dimensional cellular automaton with memory. *Journal of Physics A: Mathematical and Theoretical*, 42(48):485101.
- Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Institute Studies on the Sciences of Complexity. Oxford University Press, New York.
- Deneubourg, J.-L., Goss, S., Franks, N. R., Sendova-Franks, A., Detrain, C., and Chrétien, L. (1990). The dynamics of collective sorting robot-like ants and ant-like robots. In *Proceedings of the first international conference on simulation of adaptive behavior. From animals to animats*, pages 356–363. MIT Press, Cambridge, MA.
- Dorigo, M. and Birattari, M. (2007). Swarm Intelligence. *Scholarpedia*, 2(9):1462.
- Fatès, N. (2013). Stochastic Cellular Automata Solutions to the Density Classification Problem. When Randomness Helps Computing. *Theory of Computer Systems*, 53(2):223–242.
- Fukś, H. (1997). Solution of the density classification problem with two cellular automata rules. *Physical Review E*, 55(3):R2081–R2084.
- Fuké, H. (2002). Nondeterministic density classification with diffusive probabilistic cellular automata. *Physical Review E*, 66(6):066106.
- Gardner Jr., E. S. (2006). Exponential smoothing: The state of the art–Part II. *International Journal of Forecasting*, 22(4):637–666.

- Goss, S., Aron, S., Deneubourg, J.-L., and Pasteels, J.-M. (1989). Self-organized Shortcuts in the Argentine Ant. *Naturwissenschaften*, 76(12):579–581.
- Grassé, P.-P. (1959). La reconstruction du nid et les coordinations interindividuelles chez *Bellicositermes natalensis* et *Cubitermes sp.* La théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux*, 6(1):41–80.
- Land, M. and Belew, R. K. (1995). No Perfect Two-State Cellular Automata for Density Classification Exists. *Physical Review Letters*, 74(25):5148–5150.
- Mesbahi, M. and Egerstedt, M. (2010). Graph Theoretic Methods in Multiagent Networks. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, NJ.
- Mitchell, M., Crutchfield, J. P., and Hrabel, P. T. (1994). Evolving Cellular Automata to Perform Computations: Mechanisms and Impediments. *Physica D*, 75:361–391.
- Montes de Oca, M. A., Ferrante, E., Scheidler, A., Pinciroli, C., Birattari, M., and Dorigo, M. (2011). Majority-Rule Opinion Dynamics with Differential Latency: A Mechanism for Self-Organized Collective Decision-Making. *Swarm Intelligence*, 5(3-4):305–327.
- Montes de Oca, M. A., Ferrante, E., Scheidler, A., and Rossi, L. F. (2012). Binary Consensus via Exponential Smoothing. In Glass, K. et al., editors, Proceedings of the 2nd International Conference on Complex Sciences: Theory and Applications (COMPLEX'12), pages 244–255, Berlin, Germany. Springer.
- Rendell, L., Boyd, R., Cownden, D., Enquist, M., Eriksson, K., Feldman, M. W., Fogarty, L., Ghirlanda, S., Lillicrap, T., and Laland, K. N. (2010). Why Copy Others? Insights from the Social Learning Strategies Tournament. *Science*, 328(5975):208 213.
- Visintin, A. (2006). Mathematical Models of Hysteresis. In Bertotti, G. and Mayergoyz, I. D., editors, *The Science of Hysteresis*, volume I, chapter 1, pages 1–114. Academic Press, Oxford, UK.

Sounds Shadowing Agents Generating Audible Features from Emergent Behaviors

Insook Choi¹ and Robin Bargar¹

¹Columbia College Chicago insook@insookchoi.com

Abstract

Data from an interactive simulation of dynamic agents' social behavior is applied to the control of procedural sound synthesis. Sound is generated dynamically in parallel with visualization and presented in an interactive system where the simulation is *playable*. We discuss computational models for simulation and for procedural sound generation. Methods for extracting data from agents' behavior are applied to generate interactive sound control data. Procedural Sound Design Patterns for interactive sound are introduced with guidelines for user engagement. To illustrate the approach two sound design patterns are contrasted: (1) rendering a separate sound source for each agent, and (2) extracting swarm features from the collective behaviors of many agents to control a shared sound source.

Introduction

Dynamic agents in a simulated social environment are motivated in terms of spatial awareness. Graphical visualization reveals features from agents' emergent behaviors such as clustering, spatial pattern formation and temporal episodes. The simulation is *playable* by touching the space near agents and leading agents through social interactions on a touch-sensitive surface. The simulated social behaviors are entirely silent. We introduce procedural sound synthesis to extend agents' spatial responsiveness into audible dimensions.

Adding sound to silent moving images is common practice in interactive arts and media communications, and has also been introduced in the display of scientific data [1]. The present project is motivated by the hypothesis that sound may enhance observers' engagement by reflecting features and qualities of swarm agents' behaviors. Audible qualities of swarm data may reveal emergent features that are not easily detected in a graphical display of the same data. The present work establishes an extensible approach for investigating these types of relationship, by comparing two classes of techniques for shadowing emergent behavior with sound.

Shadowing Agents with Procedural Sound

Procedural sound generation is required to render audible features of dynamic agents' emergent behaviors. Sound computation generates audible signals by applying linear and nonlinear signal processing functions in series and in parallel at multiple timescales. The function control parameters collectively represent high dimensional control spaces where time-scaled data from dynamical system may be projected [2]. It is important to note that the swarm data is projected in the

procedural sound control space and the sound is generated in a corresponding phase space. Thus the sound procedure translates and reflects the states of the swarm system, but does not directly reproduce swarm states. Additional sound control dimensions and a unique sound generating phase space are involved in the translation of swarm data to sound.

The term *shadowing* is adopted as a metaphor to reflect a system evolving in phase space uniformly near an approximate dynamical signal trajectory [3]. Applying swarm data to control procedural sound, a "shadow" of the swarm is projected into the control dimensions of the sound generating system. As the swarm and sound systems coevolve the sound changes in ways that reflect the swarm dynamics.

This work presents two procedural sound methods that illustrate the foundations of a wide range of sound shadowing techniques spanning from data sonification [4] to automated music generation. In one method data from individual agents is applied one-to-one to control individual sound sources. In the other method feature extraction from agents' aggregate behavior elicits sound control data. Statistical methods of feature extraction enable the behaviors of large numbers of agents to control a limited number of sound sources.

Related Work

Blackwell, Unemi, Bisig, Huepe and others have applied swarm simulation to control sound computation procedures. The primary goal of most research is to generate music; in a few cases a secondary goal is to represent swarm dynamics with sound. The present authors' prior work reflects the above goals, and further explores swarm simulations as playable interfaces [5]. This research may be collectively characterized by (1) methods extracting data from the swarm simulation, and (2) methods applying data to procedural sound. Note that the selection of a type of procedural sound generator can vary widely and is not efficient as a classification of method. Note also these approaches reflect researchers' interpretation of observed swarm behaviors, and sound properties are selected a priori with respect to swarm properties, or vice versa.

Extracting Swarm Data

In prior research the prevailing approach to data extraction applies data of individual agents to represent individual sound sources [6,7,8,9,10]. This approach has been termed a *direct* [10] or *literal* mapping [11] of data to sound, to characterize the translation of spatial coordinates into sound parameter control data. The simulated perceptual relationships among

agents have no analogy in the Literal sound procedure. Each agent makes sound independently based upon absolute position in the swarm phase space, regardless of the perceptual binding of agents. A variety of sound sources have been applied to render voices of individual agents, including electroacoustic tones based upon waveform generators and signal processing [9,10], commercial synthesizers that simulate traditional musical instruments [6,7], and granular streams from prerecorded sounds [8,12]. Bisig and Neukom present a short catalog of techniques in [9]. Sound sources are differentiated by the level of control they enable, but audible emergent behavior is primarily determined by data extraction method and mapping to procedural sound control.

Other approaches generate one sound source for multiple agents. In physics-based renderings [9,10] multiple agents' interactions are applied as forces perturbing a shared sounding medium. Statistical methods for feature extraction [5] enable independent scalability of sound sources and agents. This paper compares two methods: Literal and feature extraction.

Mapping Swarm Data to Control Procedural Sound

In the research cited above the prevailing approach for mapping swarm data to sound control may be termed "fixed aural quantization of phase space." Boundaries are placed on the swarm phase space, applying modulo or reflection to agents' trajectories as needed to preserve swarms within bounds. The domain of each linear dimension is then assigned a function to rescale its position data to the range of a sound control parameter, such as frequency, amplitude, or time. An agent's absolute position in *n*-dimensional phase space defines *n* values for sound parameter control. As an agent moves in phase space its set of output values linearly covary.

A variant technique introduces a multi-swarm as a separate set of agents as attractor targets [6]. Fixed quantization of the target phase space defines the target agents' values applied to the positions of a swarm controlling sound. Blackwell applies direct manipulation to the range of the sound control mapping function to enhance the musical variety of a swarm [6].

Several approaches avoid fixed parameterization of the phase space and adopt relational interpretations based on the agents' relationships to one another rather than to phase space. Physics-based approaches (above) adopt this technique, as does a coupled-oscillator approach [10]. The authors' feature-extraction method discussed below is completely relative with respect to phase space, and individual agents' data is not directly applied to sound parameter data. Instead the transformations of relationships emerging in the swarm are applied to the transformations of relationships in sound.

Analysis of Related Methods

The present work addresses limitations inherent in the prevailing approaches: low resolution of aural quantization of swarm phase space, and limits to the number of swarm agents.

Reduction of swarm size is a significant limit of the Literal approach, where a single agent represents a single sound source. Human hearing can only distinguish a handful of independent musical voices and large numbers of similar sounds can become undifferentiated. For this reason most of the above research uses swarms of very small agent counts. Sparse swarms do not exhibit the range of emergent behaviors

characterized by greater numbers of agents. As an alternative, the relational approach discussed below enables a scalable m:n relationship between agent counts and sound sources.

Auditory downsampling of agents' data is a notable limitation of the fixed aural quantization of phase space. As the signal qualities of a sound vary continuously, audible recognition of changes and features is quantified according to non-linearity of aural perception [13] together with the audible properties of the host sound. Agents' movements can be much more detailed than a listener's aural recognition of change, and the resulting auditory quantization can generate aliasing. For example, musical patterns are based upon quantization of frequency as discrete pitches, and western listeners seek tonal orientation in stepped scales of pitches and time intervals. An agent's linear motion is inaudible between stepped pitches. We only hear change when the agents crosses a quantized frequency boundary. Likewise temporal changes are inaudible within minimum "beat" units. The effect of this relationship is that most changes at the agent level are not Quantization and aliasing can mask emergent hearable. behaviors until new features fully emerge.

As an alternative, the relational approach discussed below extracts emerging features from swarm data, then applies dynamic feature data to transform sounds. The sounds are independently designed to address perceptual orientation and to support playability of the swarm.

From Agents' Behaviors to Playability

Sayama [14] developed the swarm simulation based upon Reynold's "boids' flocking algorithm [15]. A number of agents (usually 100 to 300) are initialized at random positions in a finite virtual space and are set in motion at random initial velocities. After this initial condition, each agent's motion is dynamically influenced by its social engagement with other agents. The social engagement is simulated by assigning each agent a perceptual field defined by virtual distance units and rules for interactions with other agents. An agent's rules, known as a *recipe*, determine the agent's movement with respect to other agents located within the agent's perceptual field at the current time step:

Beyond an agent's perceptual range:

 Straying: random directional movement when the agent has no perception of others

Within an agent's perceptual range:

- Cohesion: an agent moves toward the average position of local agents
- Alignment: an agent moves towards the average velocity of local agents
- Separation: an agent avoids collision with local agents
- Whim: an agent moves randomly with a given probability
- Pace keeping: each agent approximates its speed to its own normal speed.

All agents reciprocally affect others' movements. Agents that share a recipe are called *species*. Sayama investigated heterogeneous swarms that consist of multiple species, resulting in rich emergent behaviors. Table 1 shows kinetic

parameters for each agent *i*. The pixel as distance unit has since been abstracted. Maxima were determined heuristically.

Name	Min	Max	Meaning	Unit
R^{i}	0	300	Radius of perceptual range	pixel
V_n^i	0	20	Normal speed	pixel step-1
V^{i}_{m}	0	40	Maximum speed	pixel step-1
$c^{i}{}_{I}$	0	1	Strength of cohesive force	step ⁻²
c^{i}_{2}	0	1	Strength of aligning force	step-1
c^{i}_{3}	0	100	Strength of separating force	pixel ² step ⁻²
c^{i}_{4}	0	0.5	Random steering probability	
C^{i}_{5}	0	1	Tendency of pace keeping	

Table 1: Swarm agent kinetic parameters [14]

Swarms' Salient Features and Players' Interactions

Playable interaction with swarms supports a level of observers' engagement with features emerging between visualization and sounds. Swarm agents are visualized synchronously with audio as in [16]. A superagent method enables players to interact directly with the visual display. This method leverages the existing agents' behavioral model rather than adding a separate attractor force to the simulation as in [7,8]. The superagent is a member of the swarm and perceived as a normal agent by all other agents. However, the superagent does not consult a recipe for motion and its movement is entirely determined by an external force such as a player interacting with the swarm. As the superagent moves independently the other agents respond to it as a normal agent, which empowers the superagent to exert influence on the swarm. This influence is indirect and depends on each of the recipes that are active for the other agents. Playability is thus characterized by recipes; a player cannot arbitrarily override emergent tendencies and must learn to influence the agents' behavior.

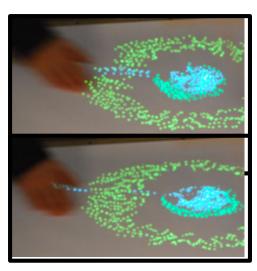


Figure 1a and 1b: A player leads a stream of agents from an encircled species cluster through an encircling species ring.

Prior to developing sound, we have set up the Sayama simulation in a playable platform in order to harvest informal descriptors of swarm behaviors from ad hoc visitors and players. Figure 1 shows a player's hand at a multi-touch

tabletop interface. The player was tasked with separating two species, visualized by color. Table 2 presents players' representative verbal expressions describing swarm agents' behaviors. Episodic descriptors and pattern descriptors portray different aspects of visible behavior. Sound shadowing may adopt criteria to consult both types of descriptors. However it is difficult to anticipate observed descriptors a priori to devise measurements, when these descriptors account for events retroactively, upon reflection after being recognized. For example, clusters are prominent emergent behavior: the spontaneous formation and dissipation of agent aggregation in subgroups (see Figure 1). When one agent "escapes" a cluster, it is a result from a mutual repelling force. However observers attribute their perception to the agent as it displays unique episodic behavior. Such attribution sets expectations for coherent explanation Sound can function in a quasiexplanatory role by providing parallel confirmation of visual events. This explanatory role is well developed in cinematic uses of sound [17].

Episodic descriptors	Pattern descriptors	
Escape	Blended	
Stray	Symmetry or asymmetry	
Common fate	Squeezed or flattened	
Exploded	Goes around and around	
Merged	Elongated or stretched	

Table 2: Players' descriptors for agents' behaviors

Based upon the observations represented in Table 2 a set of salient features was identified with quantitative measures. Some measures are endogenous features of the simulation such as agent counts and velocities. Other measures are tuned to recognize features such as clusters and events. Table 3 summarizes the data related to salient features that was captured and applied to control procedural sounds.

Generating Sounds of Agents' Behaviors

Before applying control data from swarms, a Procedural Sound Design Pattern provides preconfigured control data anticipating interactive control data. It is a unique set of preconfigured control data that creates an audible framework for interactive sound control data. When interactive control data is applied the resulting transformations maintain stable and coherent auditory properties. Criteria for sound design patterns ensure that audible characteristics are consistent and easy to identify across swarm data variations. At the same time the pattern must allow for a variety of transformations without losing coherence. Procedural flexibility accommodates unexpected combinations of data from emerging swarm behaviors. The balance of sound pattern stability and malleability is a determining factor in listeners' engagement. Below we compare two sound design patterns according to their extent of preconfigured control data and the resulting interactive responsiveness.

Preconfigured control data supporting interactive sound is analogous to data visualization techniques that establish visual field, reference frame, and iconography. A visualization pathway determines the correspondence between visual context and interactive data, and also determines the relative level of detail provided by the "frame" compared to the "data." Examples of visualization context include grid lines, textures, lighting, surface specularity [18] and parameterized icons such as glyphs [19]. These techniques require a balance of preconfigured control parameters with data driven parameters. Preconfigured data creates graphical design patterns that establish visual affordances or features to enhance the visibility of data.

Cluster-level data	Units
Number of agents	Integer
Cluster center position on screen	2 floats: X-axis, Y-axis Normalized [-1, 1] based on screen coordinates
Area covered by cluster	Integer or float;
Defined as screen area encompassed by cluster	Measured in pixels or other preferred unit
Average Agent Energy Mean value of velocity of agents in cluster	Integer; Normalized [0, 1] by scaling to max agent velocity
Cluster Velocity Velocity of center position of cluster	Float or integer
Cluster Symmetry (shape variables)	3 floats: Xsymmetry, Ysymmetry, XYsymmetry
Measurements are based upon the edges of the cluster tracking square, and ratios are measured by comparing edge lengths	 Normalized [0.1, 10] X₁ = lower square edge X₂ = upper square edge Y₁ = left square edge Y₂ = right square edge
X symmetry = ratio $X_1: X_2$	 Xsymmetry = 1: edges symmetrical Xsymmetry >1 to 10: lower edge is longer (max 10 times longer) Xsymmetry 0.1 to <1: upper edge is longer
$Ysymmetry = ratio Y_1: Y_2$	Same as X-axis data above. Ysymmetry = 1, >1, or <1
XYsymmetry = ratio of longest X edge to longest Y edge	• XYsymmetry = 1 cluster is symmetrical • XYsymmetry > 1 to 10 cluster elongated on X axis • XYsymmetry 0.1 to < 1 cluster elongated on Y axis

Table 3: Swarm and cluster data for sound control

In sound generation the balance between preconfigured and interactive control data can be analyzed along two design functions: (1) the direct relationship between sound and simulation dynamics, and (2) the indirect relationship between sound and visualization of the simulation data. Function (1)

has to do with compatibility between temporal models in sound and temporal models in simulation dynamics. The models define ranges: primarily frequency ranges including pitch and timbre (frequency spectrum), also metric ranges including beats and rhythmic patterns. Function (2) has to do with correspondences that emerge dynamically between features in sound and features in visualization. The following describes how the design process optimizes for relevant emergent features.

Applying perceptual scaling to swarm data

Two ubiquitous audible properties of sounds are their ranges and their features. These can be generated with different combinations of sound parameters such as amplitude, frequency and spectral domain (timbre), also beat and rhythm. Coherence cannot be preserved by independent linear transformation of these sound qualities. When control data is applied to transform an audible pattern, it is not just a matter of mapping raw data from a swarm agent into an audible range. Scaling of sound patterns requires transformations applied to the pattern generator not merely to individual sound parameters.

In Table 3 there are many simultaneous dynamics that might be conveyed in sound. Yet listeners cannot be expected to identify a large number of independent data-driven features. Listeners are not accustomed to sounds having meaning that expresses multivariate data. Sound design requirements optimize for coherence in sound features based upon multivariate control data. Therefor these parallel data are not used independently, they are applied in combinations to achieve an aggregate transformation of a sound pattern.

Correspondence and complement between sound and visualization. When listeners attend to sounds, audible features recognition may become differentiated from visual features recognition. For example, observers can see multiple clusters and can see wandering agents between clusters. Sound does not convey this information. When a sound is created corresponding to a cluster, listeners gain information but are unlikely to count the number of sounds to determine the corresponding number of clusters. Listeners' sonic engagement is not about counting and linear measurement. Sound from several clusters conveys differences between the clusters. In other words: sound conveys intervals and transformations, not static values.

Feature data represented in Table 3 is not acquired directly from agents' states. Feature data is derived by analyzing relationships among agents. Clusters are among the most prominent emergent features, but the simulation model does not define or persist a cluster entity. Clusters are detected by analysis of agents' aggregate positions at each time step, then cluster properties are measured. Relevant correspondences in sound depend upon cluster-level data applied to sound control.

Figure 3 illustrates a deformation sequence where one cluster is divided into two. The cluster is heterogeneous having two species forming a ring pattern. A player bringing about a cluster separation, requiring 8 to 12 seconds to complete, generated Figure 3a. Figure 3b-3d shows the sequence of separation and the corresponding statistics and histogram data, displayed in a companion diagnostic tool, with multiple histograms overlaid by color. The red bars

indicate average velocity; green indicates angle of distribution around the center; blue indicates distance from the center; and yellow and pink are average horizontal and vertical position respectively. From Figure 3b to 3d the cluster separation is reflected in the histogram of average velocity (red) and distribution angle (green). Not easily visible in Figure 3d, the distance-from-center histogram (blue) indicates the separation of species in the ring structure.

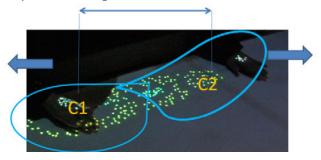


Figure 3a: Separating one cluster of agents into two

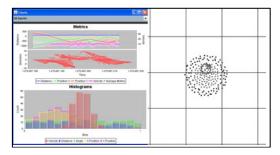


Figure 3b: Histogram of stable cluster of agents

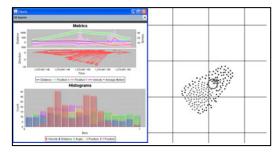


Figure 3c: As cluster deforms, the data divides

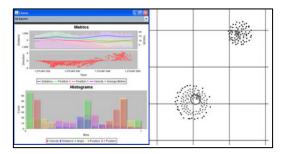


Figure 3d: Data and clusters separated

In Figures 3b and 3c the visualization includes a small circle at the center of the ring formation. This circle is a diagnostic of the cluster center as determined by the feature detection system. In Figure 3d two circles denote two clusters are detected. The following sections compare the application of feature-based data and direct agent position data as two types of procedural sound design patterns.

Procedural Sound Design Patterns: Examples

Two alternate sound design patterns were created to respond to interactive control data from the swarm simulation. The data sources were selected from the data enumerated in Table 3. As a baseline example, a literal rendition of the simulation uses minimal preconfigured control data, and each dynamic agent's position data is applied for interactive control. The other rendition applies complex preconfigured control data and also utilizes feature analysis of the simulation data. Each rendition applies a different sound synthesis design, establishing alternative audible ranges and features. In each case the sound's temporal model provides a unique level of detail reflecting the simulation's temporal characteristics. And in each case the sound's audible features establish a unique correspondence to the swarm visualization. Control rate and latency of interactive data are constant in both cases.

The present examples were developed with a normative recipe comprised of four species and 100 agents. This recipe exhibits a variety of agent behaviors with clusters forming and separating easily under players' hands. Each interpretation produces variations in sonic engagement, though these are not ranked in the present study.

Procedural Sound Setup

These categories describe both renditions of swarm-controlled sound and may be used to compare the renditions.

- a. Swarm sound control data: This is swarm data routed from the simulation to control sound.
- b. Swarm sound control parameters: These parameters apply swarm data to the audio signal generator. Note that some control parameters do not exhibit 1-to-1 correspondences to audible features of sounds.
- c. Preconfigured sound control: This sound control does not use swarm data. Preconfigured control creates a Sound Design Pattern that frames the swarm data.
- d. **Range Model:** auditory range in the Sound Design Pattern such as frequency, tempo, timbre, or rhythm relative to a data range of swarm agents' behavior
- e. **Features Model:** features in the Sound Design Pattern relative to the features of swarm agents' behavior

1: Literal Rendition

The Literal rendition creates an individual as a sound source—a "voice"—for each of 100 agents in the swarm. The accumulated instances of sound sources sustain a texture of agents' voices corresponding to their continuous movements. The result is a dense layer of simple and mostly uniform sounds that change individually without regular tempo or rhythm. This approach is similar to prior work reported above.

1.a: Swarm sound control data. The X-axis and Y-axis data for each agent at each time step are applied to control a unique sound source for each agent.

1b: Swarm sound control parameters. Sound frequency—yielding low to high pitch—is controlled by an agent's Y-axis position. Sound panning left or right in the stereo field is controlled by an agent's X-axis position.

1c: Preconfigured sound control. Two techniques are applied to generate the sound design pattern: sine wave (additive) synthesis and granular synthesis. The design pattern renders each swarm agent as a series of brief sine tones. The pitch and stereo panning of each tone are determined from the agent position (see 1a). Granular synthesis generates a series of brief tones with onset times and durations modulated by bounded randomization (see Table 4). The tone in each sound "grain" is a sine wave, with frequency determined by data from an agent's position. The durations of individual grains overlap to create a sound "texture" that is qualitatively similar to the agents' clustering behavior.

Grain duration (average)	300 milliseconds
Grain fade-in duration (avg.)	60 ms
Grain fade-out duration (avg.)	60 ms
Onset interval of grains (avg.)	100 ms
Randomization of control data	Factor in range [0.8, 1.2]
Grain loudness (amplitude)	Fixed value
Grain Frequency (pitch)	Agent data: Y-axis
Grain Stereo Panning	Agent data: X-axis

Table 4: Sound control parameters for Literal rendition

1d: Range models. The tempo of the simulation is reflected in rate of change of agents. Granular synthesis grain onset determines audible rate of change: new grains introduce new frequencies reflecting agents' position changes. A new grain onset rate of 100 ms visiting each agent of a 100-agent swarm, will sample the entire swarm at 10 Hz. The simulation calculates agents' positions at a faster rate, introducing latency in the sound compared to graphical display of agents' position changes.

Agents' X-axis positions are associated to a range of sound panning in the stereo field. Agents at extreme left or right of the graphics screen transmit sounds using only the left or right loudspeaker in a stereo configuration. Position acuity in hearing is less precise than visual position acuity.

Frequency range is tuned to reflect agents' positions on the Y-axis. Selecting the bounding frequency range is determined by *a priori* observation of agents' behavior to anticipate the likely range of cluster distribution. The human ear measures frequency change by comparing frequency ratios rather than absolute frequencies. Perceived frequencies are *pitches* and frequency ratios are *intervals*. Perceived linear pitch change requires logarithmic frequency change. To generate a linear pitch distribution we apply a logarithmic scale factor to the linear data from agents' Y-axis positions.

1e: Feature model. "Literal" is a baseline for correspondence with visual features: each agent produces an individual sound stream at a rate of ~ 10 Hz with duration of ~ 300 ms.

Persistence in sound by repetition of small multiples is compatible with agent persistence in the visual field. Given ten or more agents the outcome of this granular approach is a layered sound texture. The experiment in this rendition is to observe whether audible clusters emerge as coherent features.

1a-e Results. The Literal rendition produces perceptually ambiguous results. Agents' density and distribution is echoed in the granular synthesis sound texture, intuitively reflecting the swarm visualization. But clusters that form clearly on the screen do not emerge distinctively in sound, except when a cluster's Y-axis distribution is very narrow. As the spatial area of a cluster increases the corresponding pitch content is less differentiated from the swarm's overall frequency distribution. Clusters along the X-axis were undifferentiated. In sum, while the sound texture is generally characteristic of swarm behaviors, the sounds are not aligned to clusters and transformations that a player can generate in the visualization.

2: Feature Extraction Rendition

In the Feature Extraction rendition four sound sources are created and these respond to control data from the four largest clusters. The cluster data is updated to the sound generators at each time step. The procedural audio technique generates voices with music-based attributes: discrete tone sequences with tempo and rhythmic patterns. Data series from four clusters' transformations are applied to transform the musical timbre, tempo, and pitch attributes.

2.a: Swarm sound control data. The Feature Extraction rendition is calculated by detecting cluster formations and extracting cluster-level data. The interactive control data used in this interpretation includes: number of agents, cluster area, agent energy (a data value independent of agent velocity measured in motion), cluster position, and the XY symmetry of a cluster. The symmetry feature is an efficient method to track the continuous shape transformations exhibited by many clusters. Symmetry data is extracted from the four largest clusters and sent to four sound generators.

2b: Swarm sound control parameters. This is how the data from a single cluster is applied to sound control. The number of agents determines the maximum pitch interval between the highest and lowest pitch in the tone sequence. Smaller clusters have narrower pitch ranges. The cluster area (screen occupancy in the visualization) determines the center frequency of the sound sequence. Larger cluster areas generate overall lower pitches, regardless of the pitch bandwidth (number of agents). Agent energy determines the tempo of the tone sequence, the rate at which sound events occur. Cluster XY symmetry modulates the musical tones' timbre (tone quality or spectral structure). As cluster asymmetry increases the internal complexity of the tones increases. The tones become more complex in correspondence to the deformations of cluster symmetry. Cluster X, Y position determines panning (X-axis) and distance cues (Y-axis).

2c: Preconfigured sound control. The parameters in section 2b are applied to tone patterns that are generated by preconfigured sound control. Detailed patterns of pitch and rhythm are generated automatically using quasi-random

pattern generators that are parameterized to make music-like sequences. The swarm data is applied to produce linear timbre transformation from simple to complex. Four voices persist as part of the preconfigured sound control. However, clusters do not persist in the simulation and are unpredictable. If there are less than four clusters, the voices will be controlled by data from fewer clusters. This will result in more than one voice sharing the same cluster data from time to time. When clusters emerge or vanish the cluster-based control data is activated or deactivated at a single time step.

2d: Range model. The control data output of the preconfigured pitch and rhythm generators is scaled by the interactive data and then applied to the tone generator. The pitch and tempo ranges are scaled according to relative changes in cluster data. Highest and lowest pitches, and slowest and fastest tempos are determined with minima and maxima in cluster data, observed to *a priori*.

2e: Feature model. This cluster-analysis interpretation represents a significant shift in method. Previous methods scaled sound data based upon the dispositions of agents in simulation phase space, with the intent that emergent features will be audible through an accumulation of many brief sound events. The current method begins by extracting simulation features using targeted analysis, and then scales the sound to the attributes of the features that have been recognized. Overall this model shifts from an absolute measurement scale to a relative measurement scale.

2a-e Results. The application of feature-based data effectively generates auditory correspondences to the most notable visual changes in the clusters. The effect is such that a variety of different audio interpretations may be tested to create different emotive effects based upon the robust relationship between the visual changes and the corresponding sound changes. Players rapidly determine which clusters are controlling sounds and this ongoing discovery process becomes as aspect of engagement.

Comparison of Procedural Sound Renditions

We apply the dimensions in Table 5 to compare the sound synthesis procedures for the Literal rendition and the Feature Extraction renditions of agents' behaviors. The differences in these two approaches are characterized as "converting data to sound" vs. "using data to control sound." The Literal rendition affords "Listening to data"; the Feature Extraction rendition affords "Listening to a sound pattern transformed by data."

These two approaches can be illustrated in ratios. The Literal rendition uses a 1:1 ratio of agents to sound sources, and directly applies the data of each agent to control a single sound source. Whereas the Feature Extraction rendition greatly reduces the number of sound sources to create a ratio of agents to sound sources that varies dynamically between 10:1 and 100:1 depending upon the number of agents in each cluster. Each of four sound sources is controlled by data that statistically represents the features of a cluster, obtained across data of each agent in the cluster.

The ratio of data dimensions to procedural sound control dimensions is 1:1 in the Literal rendition. Each x-axis value

controls one audible attribute and each y-axis controls a second attribute. In the Feature Extraction rendition this ratio is 3:10. Two data dimensions from each agent are statistically summarized across as many as 100 agents, to generate data that represents features of clusters (presented in Table 3). Twenty sound control dimensions respond to six data attributes of a cluster (20:6 = 3:10).

Dimension	Literal	Feature
Number of agents	100	100
Number of sound sources (voices)	100	4
Data dimensions measured at each agent	2	2
Data dimensions measured for clusters	0	6
Num. of agents controlling each voice	1	10-100
Number of data-driven sound attributes	2	6
Frequency applied as pitch	yes	yes
Simulated directional cues	yes	yes
Simulated distance cues	no	yes
Discrete pitch sequencing and variations	no	yes
Rhythm pattern sequences and variations	no	yes
Tempo variation	no	yes
Timbre transformation	no	yes
Total number of procedural sound		
control dimensions	5	20

Table 5: Comparison of Configurations of Two Renditions

Comparing the number of data dimensions for procedural sound control shows a ratio of 1:4 between the Literal and Feature Extraction procedures. The sound palette applied in the Literal case is simplified in terms of waveform and tone onset pattern, and requires only 5 dimensions of control data. Whereas the sound palette in the Feature Extraction case is applied to differentiate multiple cluster features in sound, and uses twenty dimensions of control data.

Comparison of perceived outcomes

Related research does not include controlled user studies. Anecdotal observations are the primary source of assessment of system performance, including comments by musicians performing with swarms [6] and by exhibition visitors interacting with swarm audio installations [7, 9]. Similarly the results in Table 6 are from an informal poll of 15 players and do not represent a controlled study. The results are presented to illustrate the interplay of characteristics in the perceived relationship between visual and audible attributes of agents.

Initial assessment of the Literal rendition and Feature Extraction rendition compares a listener's experience of sounds to the experience of agents' graphical display.

- f. Pattern Definition: Compared to the agents' visual patterns, the sounds exhibit patterns that are more clearly defined, less clearly defined, or about equally defined?
- g. Episodic Behavior: The sounds exhibit episodic behavior related to agents' visual episodic behavior: always, frequently, rarely, or never?
- h. **Synchronization:** The sounds exhibit changes at time scales corresponding to agents' visual changes: always, frequently, rarely, or never?
- i. **Similarity (Isomorphism):** What degree of qualitative similarity do you find between the sounds' characteristics

- and the agents' visual characteristics: a high, medium or low degree of similarity?
- j. **Common Origin:** How often do the sounds appear to be produced by agents: always, sometimes, or never?
- k. **Certainty:** How certain are you of your responses?: 1= very confident, 0= somewhat certain, or -1= uncertain. (Item k is provided in response to items f-j.)

Outcome	Literal	k. Cert.	Feature	k. Cert.
f. Pattern Definition	Less	1	Equal	0
g. Episodic Behavior	Never	1	Freq.	1
h. Synchronization	Rarely	0	Freq.	1
i. Similarity	Med.	0	Low	1
j. Common Origin	Some	-1	Some	0

Table 6: Comparison of the Perceived Relationships of Sound and Visual Agents in Two Renditions

The relationship between the visual presentation of agents and the presentation of related sounds emerges differently with different audible renditions of agent data. Listeners' uncertainty is greater with the sounds that are not generated using a musical sound design pattern. At the same time, the sound design pattern controlled by feature extraction data exhibits a stronger association to the agents' patterns and episodic behaviors, even though the sounds are not perceived as emitted by the agents.

Interpretation and Future Work

The project aims to generate salient features from emergent behaviors, by shadowing agents' positions and movements to generate a corresponding auditory experience. Salient features are memorable landmarks that provide cognitive orientation to a full range of audio and graphic patterns. We observed that audio landmarks were not emergent as with graphical features of clusters. We determined to apply feature extraction analysis to the agents' positions and movements to automate the recognition of salient features and translate them into sound.

Feature data is applied to procedural sound to generate observable "fiducial points", aligning audible and visible behaviors as bimodal features of a coherent system relationship. The results reported here indicate that feature based data applied to sound transformations can reflect agents' emergent behaviors with fine-grained distinction. Without feature-based data, the agents' position data does not afford auditory features' correspondence to emergent visual patterns.

Future work. Formal user studies are required to validate and refine the informal findings. At this stage this project has not yet addressed agent social behavior rule selection and its impact on feature detection and procedural sound. Design of agents' behaviors in parallel to sounds and to data selection is a complex and potentially expressive practice that follows from the present investigations. Future work includes the implementation of tools for specifying and refining sounds in combination with swarm species. Creating new species can take place in an iterative design workflow coupled to a capacity to select from and develop interactive sound synthesis palettes. Also in the future conducting user

engagement studies will enable assessment of multiple sound palettes with multiple control data interpretations, in formal studies with groups of players.

References

- [1] Bargar, R., Choi, I., Das, S., and Goudeseune, C. (1994). Model-based interactive sound for an immersive virtual environment." In *Proceedings of the 1994 International Computer Music Conference*, Aarhus, Denmark.
- [2] Choi, I. (1994). Sound synthesis and composition applying time scaling to observing chaotic systems. In *Proceedings of the Second International Conference on Auditory Display, ICAD '94*, pages 79-107. Santa Fe Institute, Santa Fe, NM.
- [3] Katok, A. and Hasselblatt, B. (1995). Introduction to the Modern Theory of Dynamical Systems. Cambridge University Press, Cambridge.
- [4] Kramer, G. editor. (1994). Auditory Display. Santa Fe Institute Studies in the Sciences of Complexity, Proc. Vol. XVIII, Addison-Wesley, New York.
- [5] Choi, I., and Bargar, R. (2012). A Playable Evolutionary Interface for Performance and Social Engagement." In Camurri A. and C. Costa. C., editors, *Proceedings 4th International ICST Conference* on Intelligent Technologies for Interactive Entertainment, LNICST 78, pages 170-182. Springer, Heidelberg.
- [6] Blackwell, T. (2003). Swarm music: improvised music with multiswarms. In Artificial Intelligence and the Simulation of Behaviour. University of Wales.
- [7] Unemi, T. and Bisig, D. (2005). Music by Interaction among Two Flocking Species and Human. In Proceedings of the Third International Conference on Generative Systems in Electronic Arts, pages 171-179. Melbourne, Australia.
- [8] Davis, T. and Karamanlis, O. (2007). Gestural control of sonic swarms: Composing with grouped sound objects. In 4th Sound and Music Computing Conference, Lefkada, Greece.
- [9] Bisig, D. and Neukom, M. (2008). Swarm based computer music-towards a repertory of strategies. In GA2008, Proceedings of the 11th Generative Art Conference.
- [10] Huepe, C. Cadiz, R. F. and Colasso, M. (2012). Generating music from flocking dynamics. *American Control Conference*, pages 4339-4344.
- [11] Choi, I. and Bargar, R. (2013). Between Music and Games: Interactive Sonic Engagement with Emergent Behaviors. In Reidsma, D., Haruhiro, K., and Nijholt, A., editors, Advances in Computer Entertainment, Lecture Notes in Computer Science, Volume 8253, pages 519-523. Springer, Heidelberg.
- [12] Blackwell, T. (2007). Swarming and music. In Evolutionary Computer Music pages 194-217. Springer: London.
- [13] Zwicker, E. and Fastl, H. (2006). Psychoacoustics. Springer-Verlag, Berlin.
- [14] Sayama, H. (2007). Decentralized Control and Interactive Design Methods for Large-Scale Heterogeneous Self-organizing Swarms. In F. Almeida e Costa et al. (eds.) ECAL 2007, LNAI 4648, pp. 675–684. Berlin: Springer-Verlag.
- [15] Reynolds, C. (1987). Flocks, herds and schools: A distributed behavioral model. In SIGGRAPH '87: Proceedings of the 14th annual conference on computer graphics and interactive techniques, pages 25--34. Association for Computing Machinery.
- [16] Bisig, D., Neukom, M., and Flury, J. (2007). Interactive swarm orchestra. In *Proceedings of the Generative Art Conference*. Milano, Italy.
- [17] Chion, M. (2009). Film: A Sound Art. Columbia University Press, New York.
- [18] Ware, C. (2004). Information Visualization. Morgan Kaufman, San Francisco
- [19] Schroeder, W. and Martin, K. (2005). Overview of Visualization. In Hansen, C. and Johnson, C. editors, *The Visualization Handbook*, pages 3-35. Elsevier, Burlington, Mass.

Towards designing artificial universes for artificial agents under interaction closure

Martin Biehl, Christoph Salge and Daniel Polani

University of Hertfordshire, Hertfordshire, UK m.biehl@herts.ac.uk

Abstract

We are interested in designing artificial universes for artificial agents. We view artificial agents as networks of highlevel processes on top of of a low-level detailed-description system. We require that the high-level processes have some intrinsic explanatory power and we introduce an extension of *informational closure* namely *interaction closure* to capture this. Then we derive a method to design artificial universes in the form of finite Markov chains which exhibit high-level processes that satisfy the property of interaction closure. We also investigate control or information transfer which we see as an building block for networks representing artificial agents.

Introduction

We are interested in designing artificial physics for artificial agents. This paper presents an exploratory step in this direction and also expounds the conceptual and the formal point of view we are taking. In this introduction we give a short overview of our approach and then proceed to formally define the different elements.

Conceptually, we draw inspiration for our artificial physics and agents from "real" physics and living organisms. The artificial agents we have in mind are are minimally represented by networks of "high-level" or "macroscopic" processes. These high-level processes are derived from the underlying artificial physics. This situation is analogous to viewing living organisms as networks of processes (Maturana and Varela, 1980) on a meso- or macroscopic scale e.g. proteins or cells, and assuming an underlying physics e.g. elementary particle physics. Formally, we model our artificial physics simply as a univariate finite discrete time Markov process. We choose a univariate process because we do not want to presuppose any structure of the state space of the artificial physics. We also assume there is no downward causation (Campbell, 1974). This means that at all times, the high-level processes are causally dependent on the underlying physics. Loosely speaking, this means that the edges (interactions) of the high-level network of processes representing the agent are actually mediated by the low-level process. As we will see, this can formally be modelled using Bayesian networks.

The final ingredient of our general approach tries to account for the success of doing science on scales larger than elementary particles e.g. atomic physics, chemistry and biology. To take this into account, we require that the highlevel processes are as predictive of other high-level processes as the underlying physics itself. In other words, the high-level processes at least appear to be directly causally related. Formally, we achive this by slightly extending the notion of informational closure introduced by Bertschinger et al. (2006) to two notions that we will call *weak and strong interaction closure*. Requiring informational closure already puts some constraints on the underlying process (Pfante et al., 2014) and so do interaction closures.

Within this general setting we here inspect the situation where one high-level process seems to control another one. The idea is that any high-level network that represents an agent needs such a mechanism. Consider for example a sensor that writes its measurement to another process e.g. a memory for further processing. Another interpretation would be that the controlled process is part of the embodiment of the agent and therefore within the sphere of influence of the agent and shielded from the environment. The latter interpretation is related to the notion of embodiment put forward by Porr and Wörgötter (2005). Yet another, more conservative, interpretation would be that the first process simply transfers information to the second. Information transfer is widely seen as an important part of decentralized computation (Lizier et al., 2014). Which in turn may be just what a network of processes representing an agent needs. Formally, we use an information theoretic notion, the transfer entropy (Schreiber, 2000), to quantify (here only apparent) control. Control and transfer entropy have been linked in another context by Touchette and Lloyd (2004).

Note that the mechanism we treat is a requirement we introduce here in addition to interaction closure property. In order to arrive at a complete agent further mechanisms within larger networks are required. This will be investigated in future work.

The results in this paper show that the requirements of strong interaction closure and control from a pair of highlevel processes put strong constraints on the dynamics of the underlying process. To arrive at these constraints we assume the ideal cases of both interaction closure and control. It should be seen as an advantage of the information theoretic measures we employ that they are both "soft". This means they can readily be used to quantify also the degrees to which closure and control are present in a system.

Related work

In general, artificial agents have been studied using information theoretical concepts by several authors (e.g. Klyubin et al. (2004); Lungarella et al. (2005); Bertschinger et al. (2008); Williams and Beer (2010); Zahedi and Ay (2013)). Of those authors many also employ Bayesian networks and specifically the perception-action loop (Klyubin et al., 2004; Bertschinger et al., 2008; Zahedi et al., 2009)). The perception-action loop is a Bayesian network describing the causal relations between four stochastic processes representing environment, sensor, actuator, and memory (of the agent) states respectively. In these papers the perceptionaction loop is not seen as a network of high-level processes in our sense since the interactions between the four processes are direct and not mediated by an underlying process.

As already mentioned our notion of interaction closure is an extension of the concept of informational closure introduced by Bertschinger et al. (2006). The main difference is that we define interaction closure between two processes with respect to a third (the underlying one) while the original notion concerns closure of one process with respect to another only. We also use a stronger version of informational closure.

Conditions on underlying processes to exhibit "independence" of a high-level process from an underlying one have been studied for Markov chains at least since Kemeny and Snell (1976). They study *lumpability* which requires that the high-level process is itself a Markov process. Research in this direction has been extended in Görnerup and Jacobi (2008); Jacobi and Görnerup (2009). Very recently lumpability has been shown to be implied by informational closure by Pfante et al. (2014). In this work various other level structure measures have also been thoroughly investigated. Interactional versions were not studied though.

Our notion of apparent control or information transfer is studied in the context of distributed computation in great detail by Lizier et al. (2014). It is argued there that information transfer (measured in the same way as here) is one of three ingredients needed for computation the other two being information storage and information modification. Investigations into the computational capabilities of dynamical systems have a long history (e.g. Langton (1990); Mitchell et al. (1993) and see Lizier et al. (2014) for more). As far as we know, the focus there has not been on the implications of computation occurring on a high-level for the underlying process.



Figure 1: Bayesian network representing one time step of the relationship of the underlying process $\{X_t\}_{t\in I}$ and a high-level process $\{Y_t\}_{t\in I}$. The primed random variables represent the process state one time step after the not primed ones.

Formal concepts

Artificial universe

We start by representing an isolated system (referred to as an artificial universe or the underlying process in the following) by a finite Markov chain $\{X_t\}_{t\in I}$ on state space $\mathcal X$ defined by the time-homogenous transition kernel (or Markov matrix) $P:=p(X'|X):=(p_{x'x})$ with

$$p_{x'x} := p(x'|x) := Pr(X_{t+1} = x'|X_t = x).$$
 (1)

Our assumption is that the isolated system should be Markov, as there is no external storage of information about past states. Choosing finiteness and time discreteness is done to reduce technical issues and improve clarity of the concepts, for the same reason we restrict ourselves to the stationary case in this treatment. Stationarity may often be a valid approximation for some time interval.

High level processes

We call a random process $\{Y_t\}_{t\in I}$ on state space $\mathcal Y$ a high-level process of $\{X_t\}_{t\in I}$, if Y_t is dependent only on X_t via a transition matrix $\Pi^Y=(\pi^Y_{yx})$ defined by

$$\pi_{yx}^{Y} := \pi^{Y}(y|x) := Pr(Y_t = y|X_t = x).$$
 (2)

Note that the transitions π_{xy}^Y are independent of time. See Fig. 1 for the corresponding causal Bayesian network 2 . We also define the Bayesian inverse:

$$\pi_{xy}^{Y\dagger} = \begin{cases} 0 & \text{if } \pi^Y(y|x) = 0\\ \frac{\pi^Y(y|x) \ p(y)}{p(y)} & \text{else} \end{cases}$$
(3)

where p(x) is the stationary distribution. For a detailed investigation of high-level processes see the work of Pfante et al. (2014).

¹We choose the index set I as the integers and initialize the process in its stationary distribution at t = 0.

²Following Pearl (2000) we only draw arrows for causal interactions. Our measures on the other hand are all purely observational.

We also explicitly mention the deterministic case. Call a random process $\{Y_t\}_{t\in I}$ on state space $\mathcal Y$ a deterministic high-level process of $\{X_t\}_{t\in I}$, if $Y_t=f^Y(X_t)$ for some function $f:\mathcal X\to\mathcal Y$ we can represent such a function f^Y by a matrix $\Pi^Y=(\pi^Y_{yx})$ defined by

$$\pi_{yx}^{Y} := \pi^{Y}(y|x) = \delta_{f^{Y}(x)}(y) := \begin{cases} 1 & \text{if } f^{Y}(x) = y \\ 0 & \text{else} \end{cases}$$
 (4)

Again transitions are independent of time. The Bayesian inverse reduces to:

$$\pi_{xy}^{Y\dagger} = \begin{cases} 0 & \text{if } x \notin (f^Y)^{-1}(y) \\ \frac{p(x)}{p(y)} & \text{else} \end{cases}$$
 (5)

where

$$p(y) = \sum_{x \in (f^Y)^{-1}(y)} p(x). \tag{6}$$

Weak and strong informational closure

Informational closure was introduced by Bertschinger et al. (2006) to formalize the idea of closure known from systems theory (see references ibid.) within the framework of information theory. Loosely speaking, closure is attained by a system if it can be described without reference to the environment that it is part of (Bertschinger et al., 2006). We will distinguish between a weak and a strong form of informational closure. For a high-level process $\{Y_t\}_{t\in I}$ and underlying process $\{X_t\}_{t\in I}$ (Fig. 1) weak informational closure is defined by (see Pfante et al. (2014)):

$$I(Y':X|Y) = 0 (7)$$

where I(Y':X|Y) is the conditional mutual information. The conditional mutual information for three arbitrary random variables X,Y,Z is defined by

$$I(X:Y|Z) = \sum_{z} p(z) \sum_{x,y} p(x,y|z) \log \frac{p(x,y|z)}{p(x|z)p(y|z)}.$$
(8)

Intuitively one can read this as the amount of extra information Y contains about X that is not already in Z. So informational closure (Eq. 7) requires that the current highlevel process state Y is as predictive with respect to the next high-level process state Y' as the current underlying process state X. Note that this condition can be made stronger by requiring that Y is even as predictive of Y' as the *next* underlying process state X'. This is expressed by what we will call *strong informational closure*:

$$I(Y': X'|Y) = 0.$$
 (9)

It follows from the definition of high-level processes that strong informational closure implies weak informational closure (see Appendix A). Note that none of these conditions actually change the causal structure of the Bayesian network.

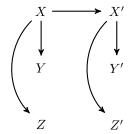


Figure 2: Bayesian network representing one time step of an underlying process $\{X_t\}_{t\in I}$ and high-level processes $\{Y_t\}_{t\in I}$ and $\{Z_t\}_{t\in I}$.

Interaction closure

We now extend the concept of strong informational closure to two high-level processes. Given two high-level processes $\{Y_t\}_{t\in I}$ and $\{Z_t\}_{t\in I}$ and an underlying process $\{X_t\}_{t\in I}$, we say that we have *strong interaction closure from* $\{Y_t\}_{t\in I}$ to $\{Z_t\}_{t\in I}$ if

$$I(Z': X'|Y) = 0.$$
 (10)

This implies (see Appendix A) the weak interaction closure:

$$I(Z':X|Y) = 0,$$
 (11)

and

$$I(Z':Y) = I(Z':X) = I(Z':X').$$
 (12)

The idea behind interaction closure is, that the states of one process are as predictive of the other's next states as the states (current or next respectively) of the underlying process.

Apparent control

In order to measure in how far one high-level process $\{Y_t\}_{t\in I}$ appears³ to control another high-level process $\{Z_t\}_{t\in I}$ we use the one-step transfer entropy

$$I(Z':Y|Z) \tag{13}$$

(Schreiber, 2000). Transfer entropy has been shown to be a measure of controllability by Touchette and Lloyd (2004). Here we say that $\{Y_t\}_{t\in I}$ appears to control $\{Z_t\}_{t\in I}$ if

$$I(Z':Y|Z) > 0.$$
 (14)

We call this apparent control because in our case the random variable Y is part of a high-level process, and does not represent a true controller. The cause of the dynamics of $\{Z_t\}_{t\in I}$ remains $\{X_t\}_{t\in I}$.

We could also use the term "information transfer" as in Lizier et al. (2014) to put more emphasis on the relation to

³Actual control would require a direct causal influence.

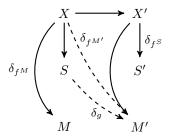


Figure 3: Bayesian network representing one time step of an underlying process $\{X_t\}_{t\in I}$ and high-level processes $\{S_t\}_{t\in I}$ and $\{M_t\}_{t\in I}$. We indicate for the case mentioned in the Result section the mechanisms associated with transitions. Dashed arrows are not part of the Bayesian network (not causal). Note δ_{f^S} is also associated with $X\to S$ and δ_{f^M} also with $X'\to M'$. This is not indicated due to space limitations.

computation, but as control was the first thing we had in mind we stick to it in this publication⁴.

Note that strong interaction closure does not imply apparent control, e.g. let $\{Y_t\}_{t\in I}=\{Z_t\}_{t\in I}$ then according to the definitions strong interaction closure implies that apparent control is zero. This is due to the fact that apparent control is based on non-causal transfer entropy and therefore a process can never (apparently) control itself.

We also use the definition of perfect apparent control (Touchette and Lloyd, 2004) to express the case where apparent control is maximal.

Perfect apparent control means for all initial states $z \in \mathcal{Z}$ and all final states $z' \in \mathcal{Z}$ there exists a state $y \in \mathcal{Y}$ such that

$$p(z'|z,y) = 1.$$
 (15)

Then I(Z':Y|Z) = H(Z'|Z) i.e. the transfer entropy attains its maximum value.

Results

Implications of interaction closure

We now present the implications of strong interaction closure for the underlying process. In order to keep the necessary technical terminology to a minimum we make a few more assumptions which lead to stronger results.

In the following we will denote the process from which the interaction closure "originates" by $\{S_t\}_{t\in I}$ and the "receiving" one by $\{M_t\}_{t\in I}$. This is done to conform to an interpretation as a sensor that (apparently) writes or transfers information to a memory. In this case strong interaction closure reads:

$$I(M': X'|S) = 0.$$
 (16)

In Appendix B. we show that under strong interaction closure and the two extra assumptions $|\mathcal{M}| = |\mathcal{S}|$ and $\{M_t\}_{t \in I}$ deterministic i.e.

$$\pi_{mx}^{M} = \delta_{f^{M}(x)}(m) \tag{17}$$

the following hold (see also Fig.3):

The process $\{S_t\}_{t\in I}$ is also deterministic with respect to $\{X_t\}_{t\in I}$ and we have an associated function $f^S:\mathcal{X}\to\mathcal{S}$. Moreover, for each $m'\in\mathcal{M}$

$$p(m'|x) = \delta_{f^{M'}(x)}(m')$$
 (18)

for some function $f^{M'}: \mathcal{X} \to \mathcal{M}$. Also for each $m' \in \mathcal{M}$

$$p(m'|s) = \delta_{q(s)}(m') \tag{19}$$

for some bijective function $g:\mathcal{S}\to\mathcal{M}$ with $g:=f^M\circ (f^S)^{-1}.$

Furthermore,

$$p(x'|x) = \begin{cases} 0 & \text{if } x' \notin (f^M)^{-1} \circ f^{M'}(x) \\ \ge 0 & \text{else,} \end{cases}$$
 (20)

and

$$\pi^{S\dagger}(x|s) = \begin{cases} 0 & \text{if } x \notin (f^{M'})^{-1} \circ g(s) \\ \ge 0 & \text{else.} \end{cases}$$
 (21)

We have thus arrived at a condition on the transition matrix of the artificial universe process from the requirement of strong interaction closure. There are two main things to take away from this.

The first is how to construct a transition matrix that obeys strong interaction closure. For this choose a finite set \mathcal{X} with $|\mathcal{X}|=n$. Then take two sets \mathcal{M} and \mathcal{S} with $|\mathcal{M}|=|\mathcal{S}|$ and functions $f^M:\mathcal{X}\to\mathcal{M}$ and $f^S:\mathcal{X}\to\mathcal{S}$. Then construct a matrix, split it vertically according to the preimages $(f^S)^{-1}$ and horizontally according to those of $(f^M)^{-1}$ (if for example the first and the last row are part of $(f^M)^{-1}(m)$ make sure to remember they belong to the same block). Make sure that each column sums to one, and note that the entries in each column can only be larger than zero in one block of the preimage of $(f^M)^{-1}$. Here is an example with $\mathcal{X}=\{1,2,3,4,5,6\}, \mathcal{M}=\mathcal{S}=\{1,2\}, f^M(x)=1$ for $x\leq 3$ else $f^M(x)=2$ and $f^S(1)=f^S(4)=1$ else $f^S(x)=2$:

$$P = \begin{pmatrix} \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & 0\\ \frac{1}{3} & 0 & 0 & \frac{1}{6} & 0 & 0\\ \frac{1}{3} & 0 & 0 & \frac{3}{6} & 0 & 0\\ 0 & \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{4} & \frac{1}{2}\\ 0 & \frac{1}{3} & \frac{1}{4} & 0 & \frac{1}{2} & 0\\ 0 & \frac{1}{3} & \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{2} \end{pmatrix}$$
 (22)

⁴Also, we don't want to discuss here what "apparent computation" would be.

The second is that we have two partitions on the state space $\mathcal X$ induced by the two functions f^M and $f^{M'}$. The former, $(f^M)^{-1}$ partitions $\mathcal X$ into blocks of states mapped to the same $m \in \mathcal M$ at the current time step and we call it the *current partition*. The latter partitions $\mathcal X$ into blocks that are mapped to the same $m \in \mathcal M$ at the next time step and we call it the *future partition*. Note that as g is bijective we can also view the future partition as induced by $f^S = g^{-1} \circ f^{M'}$ which shows that $s \in \mathcal S$ indicates the blocks of the future partition at the current time step. Note that time evolution starting in (s,m) would be $(s,m),(s',g(s)),(s'',g(s')),\ldots$ Here s',s'',\ldots are determined by the underlying dynamics.

The relation between the two partitions can take two extreme cases. The first is, when they *coincide* i.e. if for every $m \in \mathcal{M}$ exists $s \in \mathcal{S}$ such that $(f^M)^{-1}(m) \subseteq (f^S)^{-1}(s)$ and vice versa. The other extreme case is when they are *orthogonal* i.e. when for every pair $m, s \in \mathcal{M} \times \mathcal{S}$ we have $(f^M)^{-1}(m) \cap (f^S)^{-1}(s) \neq \emptyset$.

For coinciding partitions the blocks coincide and each block has unique associated high-level states $s \in \mathcal{S}$ and $m \in \mathcal{M}$. This means given s for a block, m is determined and vice versa. There is then a bijective function $h: \mathcal{S} \to \mathcal{M}$ which maps the current s to the current m (g maps it to m' the next high-level state). We can then write M = h(S) and $S = h^{-1}(M)$, the two processes up to changes of the alphabet identical.

For orthogonal partitions, in every block of the current partition there is at least one element of every block in the future partition. This means by only knowing the block of the current partition i.e. $m \in \mathcal{X}$ does not tell us anything about the current s or the next m' = q(s).

Implications of apparent control and strong interaction closure

Here we only look at implications for apparent control under the same assumptions as in the last section.

Recall that apparent control is measured in this context by I(M',S|M). We then have the current and the future partition of \mathcal{X} . We consider the two extreme cases of coinciding partitions and orthogonal partitions. For coinciding partitions, apparent control vanishes. To see this recall that we have a the bijective function h (see last section) such that

$$I(M', S|M) = I(M', h^{-1}(M)|M) = 0.$$
 (23)

To see this note that the random variable $h^{-1}(M)$ can never contain more information than M itself.

If we look at the orthogonal case we have that for every block of the current partition indicated by $m \in \mathcal{M}$ and every $m' \in \mathcal{M}$ there is an $x \in \mathcal{X}$ with $f^M(x) = m$ and $f^S(x) = m$ and g(s) = m'. But this just implies perfect apparent control, as in this case

$$p(m'|m,s) = 1.$$
 (24)

So our measure of apparent control varies from 0 to its maximum H(M'|M) due to the possible relations between the current and future partitions.

We can also ask whether perfect apparent control implies orthogonal partitions. As we need for every $m,m'\in\mathcal{M}$ an $s\in\mathcal{S}$ with

$$p(m'|m,s) = 1.$$
 (25)

we can see that in every block of the current partition corresponding to m there must be elements x in the future partition (i.e. $f^S(x) = s$) that lead to each m'. Due to strong interaction closure, and $|\mathcal{S}| = |\mathcal{M}|$ we have a one-to-one relation between m' and s given by g, so there must be elements x corresponding to each s in each block of the current partition. This means the two partitions are orthogonal.

In order to construct a transition matrix of a system with a pair of high-level processes, strong interaction closure and perfect apparent control, follow the procedure for constructing the transition matrix for strong interaction closure only. Make sure though that for each s and m there is a state $x \in (f^S)^{-1}(s) \cap (f^M)^{-1}(m)$. For example in the example of the last section with $\cap(s,m) := (f^S)^{-1}(s) \cap (f^M)^{-1}(m)$ we find $\cap(1,1) = \{1\}, \cap(1,2) = \{4\}, \cap(2,1) = \{2,3\}, \cap(2,2) = \{5,6\}$ and thus we have perfect apparent control there. We find also that, as expected, I(M',S|M) = H(M'|M) = 0.95669.

Discussion

We were looking for design principles for artificial universes especially with regard to the capability to contain artificial agents on a higher or macroscopic level. Conceptualizing artificial agents as networks of high-level processes, we focussed on the interaction of two such processes. To formalize the condition that there should be some explanatory power on the macroscopic level we introduced interaction closure as an extension to informational closure.

We found that if we require interaction closure, equal cardinalities of the high-level processes' state spaces and determinism of the receiving process, the dynamics of the underlying process must respect (see Eqs. 20, 21) two partitions of state space ⁵. How the two partitions are related is not determined by interaction closure. In other words, interaction closure does not specify the kind of interaction and requires only that it is closed with respect to the underlying process. To design an underlying process we can then choose the partitions (which induce the two processes) freely and create the transition matrix accordingly (see Results). Considering that we can choose the underlying state space arbitrarily large we expect that a large variety of high-level dynamics can be implemented in this way.

⁵The partitions also exist and are respected if the receiving process is not deterministic but the cardinality of its set of extreme points is equal to the cardinality of the other process (see Eqs. 48 and 49).

We also investigated a special kind of interaction, apparent control, between the high-level processes. It can be interpreted as one high-level process controlling the other or as one process transferring information to the other. We identified to extreme cases which occur. The first occurs if the two partitions associated with the interaction closure coincide, the two high-level process are essentially the same, and apparent control vanishes. The second occurs when the two partitions are orthogonal, the two high-level processes are complementary, and control is maximal. Intermediate relations between the partitions would led to intermediate levels of control.

In the future we want to investigate complete networks of high-level processes that are informationally and interactionally closed. Further interesting measures are the other ingredients of computation, information storage and modification as well as their localized versions (Lizier et al., 2014). These are interesting to us because computation seems relevant for artificial agents. We also want to focus on network structures relevant for artificial agents with metabolisms.

References

- Ay, N. and Polani, D. (2008). Information flows in causal networks. Advances in Complex Systems, 11(01):17– 41.
- Bertschinger, N., Olbrich, E., Ay, N., and Jost, J. (2006). Information and closure in systems theory. In *Explorations in the complexity of possible life: abstracting and synthesizing the principles of living systems Proceedings of the 7th German Workshop on Artificial Life*, pages 9–19. Jena.
- Bertschinger, N., Olbrich, E., Ay, N., and Jost, J. (2008). Autonomy: An information theoretic perspective. *Biosystems*, 91(2):331–345.
- Campbell, D. T. (1974). Downward causation in hierarchically organised biological systems. In Ayala, F. J. and Dobzhansky, T. G., editors, *Studies in the philosophy of biology: reduction and related problems*, pages 179–186. University of California Press, Berkeley.
- Görnerup, O. and Jacobi, M. N. (2008). A Method for Inferring Hierarchical Dynamics in Stochastic Processes. *Advances in Complex Systems*, 11(01):1–16.
- Jacobi, M. N. and Görnerup, O. (2009). A Spectral Method for Aggregating Variables in Linear Dynamical Systems with Application to Cellular Automata Renormalization. Advances in Complex Systems, 12(02):131– 155.
- Kemeny, J. G. and Snell, J. L. (1976). Finite Markov Chains: With a New Appendix "Generalization of a Fundamental Matrix". Springer.

- Klyubin, A., Polani, D., and Nehaniv, C. (2004). Organization of the information flow in the perception-action loop of evolved agents. In 2004 NASA/DoD Conference on Evolvable Hardware, 2004. Proceedings, pages 177–180.
- Langton, C. G. (1990). Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D: Nonlinear Phenomena*, 42(13):12–37.
- Lizier, J. T., Prokopenko, M., and Zomaya, A. Y. (2014). A framework for the local information dynamics of distributed computation in complex systems. In Prokopenko, M., editor, *Guided Self-Organization: Inception*, number 9 in Emergence, Complexity and Computation, pages 115–158. Springer Berlin Heidelberg.
- Lungarella, M., Pegors, T., Bulwinkle, D., and Sporns, O. (2005). Methods for quantifying the informational structure of sensory and motor data. *Neuroinformatics*, 3(3):243–262.
- Maturana, H. R. and Varela, F. J. (1980). *Autopoiesis and cognition: the realization of the living*. Springer.
- Mitchell, M., Hraber, P., and Crutchfield, J. P. (1993). Revisiting the edge of chaos: Evolving cellular automata to perform computations. *arXiv:adap-org/9303003*.
- Pearl, J. (2000). *Causality: models, reasoning, and inference*. Cambridge University Press.
- Pfante, O., Bertschinger, N., Olbrich, E., Ay, N., and Jost, J. (2014). Comparison between different methods of level identification. *Advances in Complex Systems*, page 1450007.
- Porr, B. and Wörgötter, F. (2005). Inside embodiment what means embodiment to radical constructivists? *Kybernetes*, 34(1/2):105–117.
- Schreiber, T. (2000). Measuring information transfer. *Physical Review Letters*, 85(2):461–464.
- Touchette, H. and Lloyd, S. (2004). Information-theoretic approach to the study of control systems. *Physica A: Statistical Mechanics and its Applications*, 331(12):140–172.
- Williams, P. L. and Beer, R. D. (2010). Information dynamics of evolved agents. In *Proceedings of the 11th international conference on Simulation of adaptive behavior: from animals to animats*, SAB'10, page 3849, Berlin, Heidelberg. Springer-Verlag.
- Zahedi, K. and Ay, N. (2013). Quantifying morphological computation. *Entropy*, 15(5):1887–1915. arXiv:1301.6975 [cs, math].

Zahedi, K., Ay, N., and Der, R. (2009). Higher coordination with less control - a result of information maximization in the sensorimotor loop. *0910.2039*.

APPENDIX

A.

To see that strong interaction closure implies weak interaction closure (snd therefore strong informational closure implies weak informational closure), note

$$I(Z':X,X'|Y) = I(Z':X'|Y) + I(Z':X|X',Y)$$
(26)

$$=0 (27)$$

where the first term on the right vanishes because it represents strong interaction closure and the second term vanishes because $\{X',Y\}$ d-separates Z' and X according to the Bayesian network in Fig. 2. In general:

$$I(Z':X|Y) \le I(Z':X,X'|Y)$$
 (28)

and as conditional mutual informations are non-negative, I(Z':X|Y)=0 as well, which means we have weak interaction closure. By replacing Z' by Y' the same argument also proves the informational closure. For d-separation in the context of Bayesian networks and conditional mutual information see Ay and Polani (2008).

To see that strong interaction closure implies

$$I(Z':Y) = I(Z':X) = I(Z':X')$$
 (29)

consider

$$I(Z':Y,X) = I(Z':Y) + I(Z':X|Y)$$
 (30)

$$= I(Z':X) + I(Z':Y|X).$$
 (31)

In both lines the second terms on the right hand side vanish. In the upper case because this is the requirement of weak interaction closure (which is implied by the strong version) and in the lower equation because X d-separates Z' and Y. This gives us the first equality in Eq.29, the second follows by replacing X by X' and using the same reasoning.

В.

Terminology and background Let $\Delta(\mathcal{A})$ denote the set of all probability distributions over \mathcal{A} . For each fixed b the conditional probability p(a|b) defines a probability for each $a \in \mathcal{A}$ and thereby an element p(A|b) in $\Delta(\mathcal{A})$. Define the convex hull C(A|B) induced by a transition matrix p(A|B) as the set of all the convex combinations of the p(A|b):

$$C(A|B) := \{ p(A) \in \Delta(A) | p(A) = \sum_{b} c_b \ p(A|b) \}$$
 (32)

here the c_b , $b \in \mathcal{B}$ are convex coefficients, i.e. for all $b \in \mathcal{B}$ we have $c_b \geq 0$ and $\sum_b c_b = 1$. Note that for deterministic

transition matrices with full rank (which we will assume in the following) $C(A|B) = \Delta(A)$.

An element e of a convex set C is called an extreme point if from $e = \sum_i c_i v_i$ with $v_i \in C, c_i > 0$ (note, strictly larger) it follows that $e = v_i$ for all i that are summed over. We denote the set of extreme points of C(A|B) by E(A|B). Note that in general for each extreme point $e \in E(A|B)$ there must exist at least one $b_e \in \mathcal{B}$ such that

$$e = p^A(A|b_e). (33)$$

Therefore $|\mathcal{B}| \geq |E(A|B)|$. In case of equality $|\mathcal{B}| = |E(A|B)|$ each p(A|b) must correspond to a different extreme point and we get a one-to-one relationship between $b \in \mathcal{B}$ and extreme points $e \in E(A|B)$:

$$p^{A}(A|b) = e_{b} \text{ and } e = p^{A}(A|b_{e}).$$
 (34)

For any probability distribution $p(A) \in \Delta(A)$ we also define the set $B_A(p(A))$ of states b with p(A|b) = p(A). Note if e is an extreme point of C(A|B) i.e. $e \in E(A|B)$ then from Eq. 33 we know that $B_A(e)$ is not empty.

In the deterministic case $p^A(a|b) := \delta_{f(b)}(a)$. The sets $B_A(\delta_i)$ for each $i \in \mathcal{A}$ then partition \mathcal{B} into $|\mathcal{A}|$ blocks and we have $B_A(\delta_i) = f^{-1}(i)$. We also have

$$\{\delta_i(A)|i\in\mathcal{A}\} = E(A|B). \tag{35}$$

Sketch of proof Now assume

- Bayesian network of Fig. 2, with $Y \to S$ and $Z \to M$,
- the stationary distribution of $\{X_t\}_{t\in I}$ has full support (for all $x\in\mathcal{X}, p(x)>0$),
- strong interaction closure I(M', X'|S) = 0,
- for each $x \in \mathcal{X}$ we have $\pi^M(M|x)$ is an extreme point of C(M|X) (e.g. if $\pi^M(M|x)$ is deterministic),
- |S| = |E(M|X)| =: k (= |M|) in the deterministic case)

A sketch of the proof is as follows.

1. First we show that

$$E(M|X) =: E(M'|X') = E(M'|X) = E(M'|S).$$
(36)

- 2. Then we show that for each $e \in E(M|X) = E(M'|X) = E(M'|S)$ the underlying dynamics p(x'|x) must map elements of $X_{M'}(e)$ into $X'_{M'}(e)$. Similarly, $\pi^{S\dagger}(x|s)$ must map elements of $S_{M'}(e)$ into $X_{M'}(e)$.
- 3. Then we prove that the sets $\{X_{M'}(e)|e\in E(M|X)\}$ and $\{X_M(e)|e\in E(M|X)\}$ are both partitions of $\mathcal X$ which induce functions $\hat f^{M'}$ and $\hat f^M$. Also that Π^S is deterministic. We then define $\hat g$. Then if Π^M is deterministic $\hat f^{M'}$ and $\hat g$ generate $f^{M'}, g$ and Eqs. 20 and 21.

Proofs

Ad 1.) Clearly, if two convex sets coincide, then their sets of extreme points coincide. So show first that

$$C(M'|S) \subseteq C(M'|X')$$
 and $C(M'|S) \supseteq C(M'|X')$ (37)

Left inclusion first:

$$p(M'|s) = \sum_{x',x} \pi^{M}(M'|x')p(x'|x)\pi^{S\dagger}(x|s)$$
 (38)

$$= \sum_{x'} \pi^{M}(M'|x')p(x'|s). \tag{39}$$

Where we only needed the Bayesian network structure of Fig. 2. So each p(M'|s) is a convex combination with coefficients p(x'|s) of the distributions $\pi^M(M'|x')$ which span C(M'|X').

Right inclusion:

$$p(M'|x') = \sum_{a,x} \frac{p(M', x', x, s)}{p(x')}$$
 (40)

$$= \sum_{a,x} \frac{p(M'|s)p(x',x|s) \ p(s)}{p(x')}$$
(41)

$$= \sum_{a} p(M'|s)p(s|x'). \tag{42}$$

Where for the step from the first to second line we used

$$p(m', x', x|s) = p(m'|s) \ p(x', x|s) \tag{43}$$

which follows directly from Eq. 26 which states:

$$I(M': X', X|S) = 0.$$
 (44)

So this time we see that all p(M'|x') are convex combinations of the p(M'|s) which proves the right inclusion.

The proof of C(M'|X) = C(M'|X') proceeds along the same lines. The sets of extreme points then also coincide i.e. Eq. 36 holds.

Ad 2.) Show that all $x \in X_{M'}(e)$ map into $X'_{M'}(e)$. We have

$$e = p(M'|x_e) = \sum_{x'} \pi^M(M'|x') \ p(x'|x_e),$$
 (45)

we see that $e=p(M'|x_e)$ is a convex combination of $\pi^M(M'|x')$ with convex coefficients $p(x'|x_e)$. But the only convex combinations that result in an extreme point have positive coefficients only for those $\pi^M(M'|x')$ with $\pi^M(M'|x')=e$ i.e. those $\pi^M(M'|x')$ with $x'\in X'_{M'}(e)$, i.e.

$$p(x'|x_e) = \begin{cases} 0 & \text{if } x' \notin X'_{M'}(e) \\ \ge 0 & \text{else,} \end{cases}$$
 (46)

which proves the condition on p(X'|X). The proof that

$$\pi^{S\dagger}(x|s) = \begin{cases} 0 & \text{if } x \notin X_{M'}(e_s) \\ \ge 0 & \text{else} \end{cases}$$
 (47)

proceeds along the same line. Notice that each $s \in \mathcal{S}$ is an s_e (Eq. 34) and we therefore moved the index in Eq. 47.

Ad 3.) $\{X_{M'}(e)|e\in E(M|X)\}$ is a partition iff a.) for $e_1\neq e_2\in E(M|X)$ $X_{M'}(e_1)$ and $X_{M'}(e_2)$ are disjoint and b.) for all $x\in\mathcal{X}$ there exists $e\in E(M|X)$ with $e\in X_{M'}(e)$. Note a.) is true by construction. We show b.). Take an arbitrary $x^*\in\mathcal{X}$. Notice that there exists $s^*\in\mathcal{S}$ with $\pi^S(s^*|x^*)>0$ because Π^S has full rank. But then via definition (Eq.3) and using that p(X) has full support we get $\pi^{S\dagger}(x^*|s^*)>0$. But Eq. 47 tells us that then $x^*\in X_{M'}(e)$ for some unique e. This means for every $x^*\in\mathcal{X}$ there is e_{x^*} with $x^*\in X_{M'}(e_{x^*})$. This proofs b.) and allows us to define a function $\hat{f}^{M'}:\mathcal{X}\to E(M|X)$ via $\hat{f}^{M'}(x^*)=e_{x^*}$. Then $(\hat{f}^{M'})^{-1}(e)=X_{M'}(e)$.

Next show that $\{X_M(e)|e\in E(M|X)\}$ is a partition. Recall $X_M(e)=X'_{M'}(e)$ because of time independence of the high-level processes. Again disjointness is clear. Notice that because the underlying process is positive recurrent (as it has a stationary distribution) there exists $x\in\mathcal{X}$ with $p(x'^*|x)>0$. Then from Eq. 48 there must exists a unique e with $x'^*\in X'_{M'}(e)$. So $\{X_M(e)|e\in E(M|X)\}$ is also a partition and we define the function \hat{f}^M analogous to $\hat{f}^{M'}$. We can now extend Eq. 46 and get:

$$p(x'|x) = \begin{cases} 0 & \text{if } x' \notin X'_{M'}(e_x) = (\hat{f}^M)^{-1} \circ \hat{f}^{M'}(x) \\ \geq 0 & \text{else.} \end{cases}$$

Now show that Π^S is deterministic. Let $s_1 \neq s_2$ and $\pi^S(s_1|x), \pi^S(s_2|x) > 0$. This implies $\pi^{S\dagger}(x|s_1), \pi^{S\dagger}(x|s_2) > 0$ and from Eq.47 $x \in X_{M'}(e_{s_1})$ and $x \in X_{M'}(e_{s_2})$ which implies (disjointness) $e_{s_1} = e_{s_2}$ which is not possible as $|\mathcal{S}| = |E(M|S)|$ (see Eq. 34). We then have an associated function $f^S: \mathcal{X} \to \mathcal{S}$.

Define $\hat{g}:=\hat{f}^{M'}\circ (f^S)^{-1}$ (it is bijective). Then

$$\pi^{S\dagger}(x|s) = \begin{cases} 0 & \text{if } x \notin X_{M'}(e_s) = (\hat{f}^{M'})^{-1} \circ \hat{g}(s) \\ \geq 0 & \text{else} \end{cases}$$

$$\tag{49}$$

If Π^M is deterministic, $e = \delta_i$ ($i \in \mathcal{M}$ see Eq. 35) and we define $f^{M'}$ and g by requiring: if $\hat{f}^{M'}(x) = e = \delta_i$ then $f^{M'}(x) := i$ and if $\hat{g}(s) = e = \delta_i$ then g(s) = i. If this is plugged into Eqs. 48 and 49 we get Eqs. 20 and 21. End of proof.

Framework for Adaptive Swarms Simulation and Optimization using MapReduce

Sergi Canyameres¹, Doina Logofătu²

¹²Computer Science Department of Frankfurt am Main University of Applied Sciences, 1 Nibelungenplatz 60318, Frankfurt am Main, Germany logofatu@fb2.fh-frankfurt.de

Abstract

Natural swarms can have multiple structures and follow many different patterns or laws. Most recent studies try to obtain the best solutions in very specific and sophisticated contexts. In this research, instead, an application to find a reasonable approximation to the behaviour of any possible incoming environment is developed. This paper describes a basic framework, GUI and algorithms established in a way that can be easily modified and adapted to desired paradigms, parameters and rules. In experimental research, this will be able to provide help for many applications where natural swarm patterns are followed but no deep, expert simulator has yet been developed. To deal with the complexity of the biggest applications, the simulations are to be run in parallel computing using the MapReduce framework.

Background and Motivation

We were asked to develop a software which would fit the requirements for taking part in the InformatiCUP [1]. In this year's edition the participants were asked to find the best algorithms some robots should follow under a specific scenario [2]. These are deployed on the ocean's surface and can only communicate between other nearby boids, knowing not more than their relative positions. The aim is to move around the ocean's surface collecting manganese and gather together after a certain time or command. Many spreading or path algorithms could be appropriate, but there was a lack of information about what to optimize (time, fuel, unique track covered ...) so the goals were too ambiguous. None of the existing swarm algorithms that we found satisfied our expectations. As seen in previous studies [3][4][5], it's common to focus research in carefully delimited conditions, but no adaptable platform to use with our constrictions was found.

The goal of our project switched to cover this lack of resources and try to set the basis for a lot of possible future developments and applications in experimental research. Different states and phases were introduced to modify the

different algorithms' weights according to the desired behaviour of the robots at every moment.

Section 2 thoroughly explains the context of the project and the first steps taken, describing how starting with different deployment patterns causes a need of very different movement rules, so regular shapes -square, circle-and irregular deployments -random, Gaussian, combined-were created to study all the possibilities. In section 3, we show how the application itself works, and how the iteration over the desired parameters can provide some acceptable results and how should they be understood for future applications. Section 4 introduces the further need of using parallel computing for the execution of the application. In section 5 the results are showed and commented to extract some conclusions leading to the future work purposed in section 6.

Requirements and Basis Settled

The requirements for the InformatiCUP were constrictive, especially when establishing which communications could take place between the robots. The uncertainty caused by the incomprehensibly ambiguous formulation of the task gave us total freedom to focus on the creation of a new versatile framework and leave constrictions behind. The required small viewing range and limited knowledge lead to the idea of applying swarm particle algorithms. Any other paradigm followed would be by intuition, and the time was limited to a few weeks which could not be spent on trying unfunded ideas.

To keep the difference between what the robots could know and what not, a Simulator class was created to keep track and manage most of the information and share it properly with the other classes. The robot instances are stored in a *HashMap* also containing information about position (classes Robot and Coordinates). Simultaneously, a first visualisation of a "sea" of 500 by 500 positions was

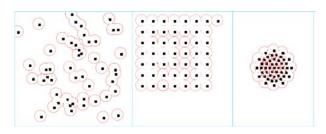
implemented, so it was time to start with the deployment configurations.

Optimising the gathering time is different to optimising the unique surface travelled or the fuel used after they start. Moreover, if the initial set of robots is positioned in different way, it becomes a key fact when it matters of which algorithm to follow in order to run over the surface following some criteria.. Hence, many deployment methods are available:

DeployRandom(). Deploys an amount of robots in random positions within the given sea limits. Initially this algorithm was implemented to verify and debug the need of creating a new robot only within the viewing range from at least one other robot. Later in the project, this distribution stayed in the simulator for further testing of new features (new visualisation, determine movement behaviours) and as one of the starting point algorithms for the InformatiCUP. As it is a very inefficient starting set the challenge was more ambitious.

DeploySquare() and **DeployCircle().** Both deploy an amount of robots following uniformly filled squared or circle patterns. The first robot is one of the four central points of the figure, which grows in a spiral or surrounding shape from the centre point.

DeployGauss() and **DeployBadCenters()**. Given the initial random centre (mean) and variance, deploy the robots following a Gaussian distribution, or in two unequal Gaussian distributions. This offers an interesting and challenging beginning, where two unequal groups are deployed close enough to see each other but far enough to, following the main algorithms seen later, tend to split up into two groups of boids. This is very useful when extreme conditions are going to be tested.



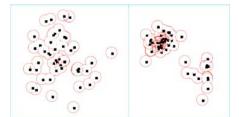


Fig. 1. In order, Random, Square, Circle, Gauss and BadCenters deployments shown in the GUI.

Previous Work

3.1 Interaction Simulator-Robot

The Simulator coordinates the evolution of the steps taken by the robots while interacting with them in order to provide the exclusive data that they are allowed to know, in the real way robots would perceive their environment. For example, the absolute coordinates are translated to relative coordinates for every robot prior to being given. Therefore, the boids are just computing machines which translate the given inputs into new positions, given a set of rules or behaviour algorithms which can be easily exchanged.

Iteration can be understood as a real-time second, fraction, etc. and the maximum speed can be changed as a consequence of this. In every step, the movement for all the robots is calculated, and only once they have found their new position they will actually move, the Simulator being responsible to iterate on the robots asking for the displacement vector. With this, both new positions and speed can be stored in order to be used in the following iterations as well as displayed through a friendly GUI.

This way of interacting is possibly not as easy to modify as the algorithms that the robots are going to use, as it is something quite specific for our project. Due to the expected versatility for the future uses, the complete code is carefully documented with Javadoc in order to make it more understandable and accessible.

3.2 Used Algorithms

Many different algorithms for moving objects or swarm behaviour could be implemented [6][7]. As said before, the main intention is not to simulate some specific scenario, but to create a platform which allows an easy transformation of the characteristics to simulate. For this reason the algorithms to be followed are not required to be very complex, only implemented in a strict modular way. This allows the addition of other algorithms or simply the modification of the existing ones.

The current application runs under a simplification of the bird flock movement described by Craig W. Reynolds [8]. This behaviour paradigm consists of three criteria every robot follows at each iteration, which can be understood as three different algorithms simultaneously working.

Cohesion: Every robot moves toward the centre of mass of the neighbouring ones (1). We assume the neighbouring group as the set of robots given by the simulator. This means that these neighbours are only those close enough (*view range* limit) to be detected by the robot. Although (1) is the concept simplification, experience made us add a

regulator which increases the value of this result if a robot is too far away from the others.

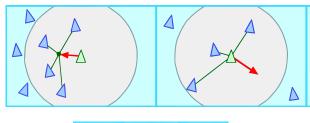
alg l(neighbours 1...n) =
$$\frac{1}{n} \sum_{i=1}^{n} (r_{ix}, r_{iy})$$
 (1)

Separation: Every robot tries to keep a minimum distance with the closest robots (2). In the same way as the first algorithm, the average position of the robots plays a role in this calculation, but a little variation gives an extra value to this result if the robot is too close from another one. The main difference is that only the really close robots are taken into consideration in this calculation. Otherwise, it would be complementary, opposite to (1) and it would not make sense to keep two algorithms.

alg2(neighbours 1...n) =
$$-\frac{1}{n}\sum_{i=1}^{n}(r_{ix}, r_{iy})$$
 (2)

Alignment: Every robot changes direction to the average position where the other robots are trying to steer to (3). This is similar to an inertial force influencing all the set of boids, as it does not work with the positions, but with the velocities.

$$alg3(velocities 1...n) = \frac{1}{n} \sum_{i=1}^{n} (v_{ix}, v_{iy})$$
 (3)



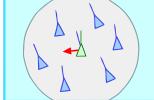


Fig. 2. Representation of algorithms (1), (2) and (3) [8].

3.3 Weights and Gathering mode

The final decision of every robot is conditioned by different weights given to every algorithm implemented (4). This allows activation, modification and deactivation of any existing algorithms without spending time programming or changing the settings. In fact, the GUI itself provides glide bars which allow the direct setting of the parameters within a given range. The current version

only applies the changes prior to the deployment, but in the future can easily be changed.

$$v(algorithms \ 1...n) = \frac{1}{n} \sum_{i=1}^{n} w_i \ alg_i(x, y)$$
 (4)

In case of exceeding the maximum speed allowed by the requirements, both x and y components can be normalized so that the vector speed fits the specifications:

$$\bar{v}[x, y] = maxSpeed * \left[\left(\frac{x}{\sqrt{x^2 + y^2}} \right), \left(\frac{y}{\sqrt{x^2 + y^2}} \right) \right]$$
 (5)

The original requirements asked for a final behaviour in which all the robots should stop focusing on collecting manganese and moving towards some common point where a mother ship could collect them. Again, the criteria to look for this point was not specified, and many possibilities could be optimized: time, fuel used... The best point if the desired parameter to minimize is the total fuel consumed (or simply the total distance travelled), this should be the average position of all the robots. Hence, the existent weights used for the algorithms are adequately used for this final situation, and all of them are set to zero except the cohesion (1) weight, which is set to the maximum value. In the basic performance, a maximum of iterations can be set in order to activate Gathering Mode, simulating an eventual call from the mother ship or simply a time limit which could be integrated in the boids. Another countdown is carried on by checking if the robots moved more than a specified threshold. After a certain number of iterations (consecutive or not) when the robots are moving less than this limit set, the Gathering Mode can also be activated. Once the meeting of all the robots has taken place, the simulation stops after a specified timer and a reset of all the variables and parameters is done, in order to execute a new simulation. In case of looking for the best parameters (see next paragraph) the initial deployment positions are loaded again.

3.4 Find Best Parameters

The most important function is responsible for simulating different executions by using as many parameters modifications as possible in order to find the best configuration. The results can be understood in many ways, as there are many output values which could be optimised (manganese collected, distance travelled, ratio between manganese and distance, iterations to join, etc.). In our case the chosen one is simply the maximum amount of minerals collected.

The most logical parameters to iterate on are the weights. However, also the number of deployed robots, the deployment method or the iterations until gathering plays an important role on the results. In any case, the simulation

ends up with a generated data file containing a header with general information about the simulations; static parameters or simply sequence of parameters used, e.g. simulation done with 5, 10, 50 and 100 robots. All the results follow the header, with particular info about the set of variables for each simulation, and finally the results for manganese collected and distance travelled.

Initially only the best configuration for the same set of parameters was registered. However, in order to understand the effects and the evolution of the values better, the output file now includes the result for all the simulations. Its manipulation and understanding would be too tedious to be done manually, so a script in Matlab is created to read and plot the results in a 3D graphic, which is very helpful to see the behaviour of the simulations. As some simulations may have more dimensions, unable to be shown in 3D, some adaptations may be done. For example, (1) and (2) could be understood as complementary values (the real difference is explained in section 3.2). Instead of using an axis for each, a new parameter can be shown as the ratio between these two values given a closed range:

$$w_1, w_2 \in [0, 4] \Rightarrow w_{12} = \frac{w_1}{w_2}$$
 (6)

3.5 Double Deployment Comparison

When finding the best parameters for a given configuration, the actual best weights might not follow an intuitive pattern which allows easy understanding. The analysis of the simulations so far was mostly a posteriori, by studying and searching a proper interpretation of the results given in the output files or the graphics in Matlab. The visualisation of the simulation for a specific set of parameters was implemented since the very beginning, but the weights were not modifiable and the proper analysis was not comfortable. Hence, we wanted to offer a functionality which could allow the user to observe and compare the behaviour of the flock in a practical, real-time way. For that, a second deployment can take place simultaneously, shown in another colour. The parameters, number of robots and the starting distribution can be different between them, offering the possibility to see in real time the development of both simulations in parallel. Fig.3 shows the visualisation offered by the GUI. On the left, a first deployment in red follows the double-Gaussian pattern, while the blue one is purely random. On the right side, a fulfilled circle is drawn by the red robots, whereas a different amount of robots, in blue, create a square.

Both double simulations perform a similar evolution, although the uniform deployments experiment smoother modifications due to the regular distance between the boids. On the other hand, the randomly positioned robots need more iterations to find a balance, which still look more irregular than in the images on the right. It was interesting to add this little option in order to explore the

capabilities of the platform in the gaming world, if we understand the both flocks as a competition to see who gets a better performance. In the future, other applications might be interested in using this second deployment with newer functionalities, such as direct modification of the parameters, commands on splitting or merging the flocks, and countless other possibilities which can be added.

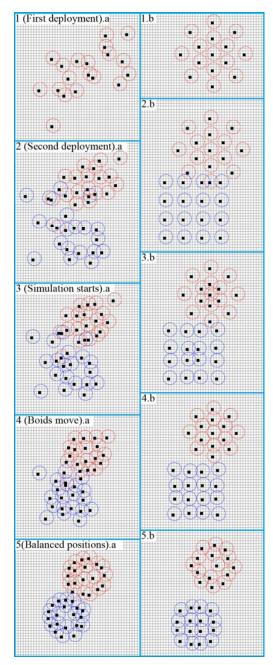


Fig. 3. Visualisation of two pairs of parallel simulations

Distributed Algorithm Using MapReduce

Nowadays it is quite strange to find big simulations being run in simple computers, and this project may not be an exception. Most of the simulations done so far by us have used low amount of robots and iterations, as well as some simplifications to just confirm the operative power of the system. In order to execute these simulations in a current generation pc, the algorithms can be sophisticated but the parameters must be simplified when finding the best configurations. Otherwise too much memory usage is needed and the program's behaviour can be irregular due to exceed of the memory space.

For all this, an implementation of algorithm parallelization is required to be the next step in the project. The Hadoop [9] open-source implementation of the MapReduce [10] model is likely to be successfully used, like in other evolutionary applications [11]. This is the OpenSource MapReduce framework implementation from Apache, a batch data processing system for running applications, which process vast amounts of data in parallel, in a reliable and fault-tolerant manner on large clusters of compute nodes, usually running on commodity hardware. It comes with status and monitoring tools and offers a clean abstraction model for programming, supporting automatic parallelization and distribution. Hadoop comes with a distributed file system (HDFS) that creates multiple replicas of data blocks and distributes them on compute nodes throughout the cluster to enable reliable, extremely rapid computations.

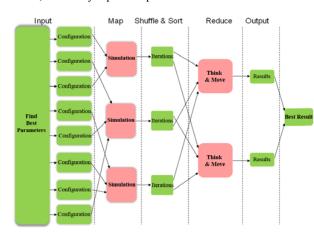


Fig. 4. MapReduce Dataflow

Initialize(Deployment)

```
def MR_MAP(List of Array possibleParameters)
= {
    for( i ← 1; i ≤ combinations; step 1)
        settings[i] ← generateConfiguration();
    end for }
```

```
def MR_REDUCE (List of <Configurations>
settings) = {
  for( i ← 1; i ≤ settings.size(); step 1)
    for( i ← 1; i ≤ settings[i].iterations; step 1)
neighboursList[i] ← getNeighbours(robots,
    coordinates);
nextMove ← robot.move(neighboursList[i]);
coordinates.update(nextMove);
   end for
  end for
}
return bestWeights;
```

Fig. 5. Pseudocode for MR_SW_OPT

The compute and storage nodes are typically the same. This allows the framework to schedule tasks effectively on the nodes where data is already present, resulting in very high aggregate rate across the cluster. The framework consists of a single master JobTracker and one slave TaskTracker per compute node. The master is responsible for scheduling the tasks for the map- and reduce-operations on the slaves, monitoring them and rerun the failed tasks. The slaves execute the tasks, as directed by the master. The applications specify the input/output locations, supply map and reduce functions and possibly invariant (contextual) data. These comprise the job configuration. The Hadoop job client then submits the job (Java byte code packed in a jar-archive) and configuration to the JobTracker, which then distributes them to the slaves, schedules the map-/reduce- tasks, and monitors them, providing status and diagnostic information to the job client.

A MapReduce job splits the input data into independent chunks (splits), which are then processed by the map tasks in a completely parallel manner. The framework sorts the maps' outputs and forwards them as input to the reduce tasks.

Experimental Results

Assuming the limitations of executing the application in a single domestic computer, different simulations were done. The initial goals were just to check the correct behaviour of the application. For this, logical results were pursuit by using understandable and basic parameters. Some combinations hold useless values. A perfect circle consisting on a regular amount of robots may not even experience any variation of the positions, as the balance is achieved since the deployment itself. Similar behaviours may occur with a perfect squared pattern, where the inner robots are balanced and only the surrounding ones experience some little repositioning. In any case, the events seen were correct. The actual implementation of the described project can be found at [12]. Once this was verified more extensive calculations took place by

increasing the range of possible values. More real-like weights were introduced. Also longer simulations could be held by deploying more robots, as well as trying all the possible deployment methods.

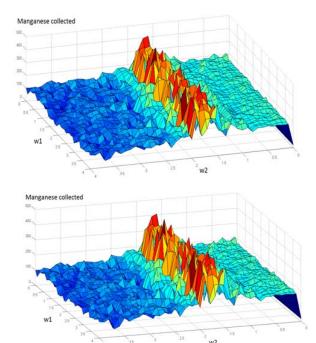


Fig. 6. Manganese depending on w_1 and w_2 after Random (above) and Square (bottom) deployments.

As the figures show, the evolution of the Manganese recollection follows a logical and acceptable evolution through the values for the main weights for algorithms (1) and (2) after 25 and 100 iterations respectively. These two graphics corresponds to two very different deployments such as random and squared-pattern. Despite the initial variations both of them look very similar. This can suggest that the robots act individually, without noticing a big influence from the surroundings. In the same way, circular and random deployments offer very similar outputs.

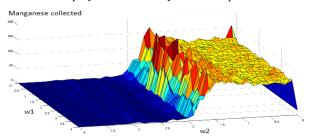


Fig. 7. Bigger simulations need more memory.

However, more detailed and long simulations should still be done. The precision is increased and the parameters are tested with smaller steps between simulations, causing a big increase of the data generated needed to be stored. Unfortunately, a standard 6Gb computer experiences difficulties to keep such volumes of information, leading into unstable behaviour of the program as seen in figure 6. It is to be understood that the output is suddenly decreased until 0, which is a senseless value because the robots gather some manganese even if they are static. In any case, the performance seen so far confirms the viability of using the framework to experiment and research with further contexts, using new algorithms. In conclusion, there are good expectations for the future.

References

- [1] InformatiCUP. http://informaticup.gi.de
- [2] Detailed requirements for the first prototype. http://informaticup.gi.de/fileadmin/reda ktion/Informatiktage/studwett/Aufgabe_Ma nganernte_.pdf
- [3] Fernandes, C.M., Merelo, J.J., Rosa, A.C.: Controlling the Parameters of the Particle Swarm Optimization with a Self-Organized Criticality Model. PPSN XII (II) pp. 153-164. Springer, Taormina (2012)
- [4] Bim, J., Karafotias, G., Smit, S.K., Eiben, A.E., Haasdijk, E,.: It's Fate: A Self-Organising Evolutionary Algorithm. PPSN XII (II) pp. 185-194. Springer, Taormina (2012)
- [5] McNabb A., Seppi, K.: The Apiary Topology: Emergent Behavior in Communities of Particle Swarms. PPSN XII (II) pp. 164-173. Springer, Taormina (2012)
- [6] Rodriguez, F.J., García-Martínez, C.: An Artificial Bee Colony Algorithm for the Unrelated Parallel Machines Scheduling Problem. PPSN XII (II) pp. 143-152. Springer, Taormina (2012)
- [7] Montes de Oca, M.A.: Particle Swarm Optimization -Introduction. http://iridia.ulb.ac.be/~mmontes/slidesC IL/slides.pdf
- [8] Reynolds, C.: Boids (simulated flocking). http://www.red3d.com/cwr/boids/
- [9] Apache Hadoop open-source implementation. http://hadoop.apache.org/
- [10] Dean, J., Ghemawat, S: MapReduce: simplified data processing on large clusters, In: Communications of the ACM, Vol. 51, Nr. 1, pp. 107-113, ACM New York, USA (2008)
- [11] Logofătu, D., Dumitrescu, D.: Parallel Evolutionary Approach of Compaction Problem Using MapReduce, In: Proceedings of 11th International Conference on Parallel Problem Solving from Nature (PPSN(2) 2010), LNCS 6239, pp. 361-370 (2010)
- [12] Implementation Adaptive Swarm Optimization (Robots): http://en.file-upload.net/download-8770063/Simulator.jar.htm

In Vitro Reconstruction of Functional Membrane

Yutetsu Kuruma^{1,2}, Hideaki Matsubayashi², and Takuya Ueda²

¹Earth-Life Science Institute, Tokyo Institute of Technology ²Gradient School of Frontier Sciences, The University of Tokyo kuruma@elsi.jp

Abstract

One feasible strategy in constructing an artificial cell is the assembly of biomolecules to fulfill the basic cellular reactions which necessary for cell alive (Luisi et al., 2006). In this strategy, membrane vesicle has been widely employed as a model cell compartment allowing internal biochemical reaction such as gene expression (Fujii et al., 2013), lipid biosynthesis (Kuruma et al., 2009), and so on. Although difficulties constructing biochemically in functional membrane composed of lipids and membrane proteins have limited the construction of a viable artificial cell, it has been reported that some kind of membrane protein can be spontaneously integrated into a lipid membrane during the translation reaction by ribosome (Ashley et al., 2012). This phenomenon can be efficiently applied for the construction of membrane protein architecture on the vesicle membrane. On the other hand, not all membrane proteins can spontaneously integrate in the native-like conformation. For example, if a membrane protein contains a large hydrophilic domain at the other side of membrane, this type protein requires a membrane translocation channel, which called as Sec Translocon (Luirink et al., 2005). Therefore, if the Sec translocon could be synthesized within the vesicle, any membrane protein should be synthesized as downstream reaction in the regulated membrane integration. In either spontaneous and Sec dependent membrane integrations, the important point is that membrane proteins have to be synthesized in the presence of vesicles in totally artificial way. In this sense, in vitro protein synthesis system, called as "cellfree system", is used as an underlying technology (Shimizu et al., 2006). A cell-free system enables protein synthesis by just adding an interest gene into the cell-free reaction mixture. Thus, the cell-free system encapsulated in vesicles can synthesize membrane proteins inside and the synthesized membrane proteins are spontaneously localized onto the vesicle membrane. This is somehow similar to what a real living cell is doing. So far, we have synthesized several membrane enzymes using the technologies of cell-free system and vesicle manipulation (Kuruma et al., 2009; Kuruma et al., 2005; Ozaki et al., 2008; Kuruma et al., 2012). For instance, two membrane enzymes, which involved in phospholipids biosynthesis pathway, synthesized within vesicles produced new lipid molecule (Kuruma et al., 2009). The membrane localization of such internally synthesized membrane protein can be visualized under the microscopy observation when a

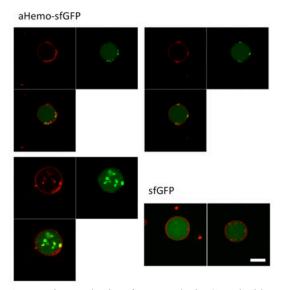


Fig. 1. Protein synthesis of α Hemolysin-GFP inside giant unilameller vesicle.

tandemly conjugated α Hemolysin-GFP chimeric protein was synthesized in giant unilameller vesicles (Fig. 1). This method will provide new aspects in artificial cell study and dynamic analysis of membrane protein (Shimizu et al., 2014).

Addition to the internal protein synthesis approach, reconstruction of biological reactions involved around vesicle membrane is also important for the development of artificial cell, and also for the deep understanding of the basic structural and dynamic organization of living cells. So far, cell division machinery (Osawa et al., 2008) or cytoskeleton structure (Maeda et al., 2012) has been partially reconstructed on the vesicle membrane. Although these membrane functions are important, self-production of ATP is crucial for autonomous cell alive. About this point, we have reconstructed an artificial which consists of ATP synthase bacteriorhodopsin (bR). bR is a proton pump membrane machinery stimulated by irradiation of light. The generated proton gradient between outside and inside of vesicle drives the ATP synthase that was also integrated in the vesicle membrane. We have reconstructed this cell-like device and succeeded to detect ATP production depending on light irradiation.

We would like to present recent results on the construction of artificial cell by means of biomolecules and membrane vesicles, focusing on cell membrane functions.

References

- Luisi, P. L., Ferri, F. & Stano, P. (2006) Approaches to semi-synthetic minimal cells: a review. Die Naturwissenschaften 93:113.
- Satoshi Fujii, Tomoaki Matsuura, Takeshi Sunami, Yasuaki Kazuta, Tetsuya Yomo.(2013) In vitro evolution of hemolysin using a liposome display. PNAS.,110:16796–16801.
- Kuruma, Y., Stano, P., Ueda, T. & Luisi, P. L. (2009) A synthetic biology approach to the construction of membrane proteins in semisynthetic minimal cells. *Biochim Biophys Acta* 1788:567–74.
- Ashley, R. L., Catherine, C. O. & Nathan N. A. (2012) The Cell-Free Integration of a Polytopic Mitochondrial Membrane Protein into Liposomes Occurs Cotranslationally and in a Lipid-Dependent Manner. PLoS One 7(9): e46332.
- Luirink, J., von Heijne, G., Houben, E. & de Gier, J. W. (2005) Biogenesis of inner membrane proteins in Escherichia coli. *Annu Rev Microbiol* 59:329–55. Review.
- Shimizu, Y., Kuruma, Y., Ying, B. W., Umekage, S. & Ueda, T. (2006) Cell-free translation systems for protein engineering. FEBS J. 273:4133–40.
- Kuruma, Y., Nishiyama, K., Shimizu, Y., Müller, M. & Ueda, T. (2005) Development of a minimal cell-free translation system for the synthesis of presecretory and integral membrane proteins. *Biotechnol Prog.* 21:1243–51.
- Ozaki, Y., Suzuki, T., Kuruma, Y., Ueda, T. & Yoshida, M. (2008) UncI protein can mediate ring-assembly of c-subunits of FoF1-ATP synthase in vitro. *Biochem Biophys Res Commun.* 367:663–6.
- Kuruma, Y., Suzuki, T., Ono, S., Yoshida, M. & Ueda, T. (2012) Functional analysis of membranous Fo-a subunit of F1Fo-ATP synthase by in vitro protein synthesis. *Biochem J.* 442:631–8.
- Shimizu, Y., Kuruma, Y., Kanamori, T. & Ueda, T. (2014) The PURE system for protein production. *Methods Mol Biol.* 1118:275–84.
- Osawa, M., Anderson, D. E. & Erickson, H. P. (2008) Reconstitution of contractile FtsZ rings in liposomes. Science 320:792-4.
- Maeda, Y. T. et al. (2012) Assembly of MreB Filaments on Liposome Membranes: A Synthetic Biology Approach. Acs Synth Biol 1:53-59.

Evolving spiking neural networks for temporal pattern recognition in the presence of noise

Ahmed Abdelmotaleb^{1,2}, Neil Davey¹, Maria Schilstra¹, Volker Steuber¹, Borys Wróbel^{2,3}

¹Biocomputation Research Group, University of Hertfordshire, Hatfield, Herts AL10 9AB, UK
²Evolutionary Systems Laboratory, Faculty of Biology, Adam Mickiewicz University, Poznań, 61-712, Poland
²Systems Modeling Laboratory, IO PAS, Sopot, 81-712, Poland

a.m.abdelmotaleb@herts.ac.uk, n.davey@herts.ac.uk, m.j.1.schilstra@herts.ac.uk, v.steuber@herts.ac.uk, wrobel@evosys.org

Abstract

Nervous systems of biological organisms use temporal patterns of spikes to encode sensory input, but the mechanisms that underlie the recognition of such patterns are unclear. In the present work, we explore how networks of spiking neurons can be evolved to recognize temporal input patterns without being able to adjust signal conduction delays. We evolve the networks with GReaNs, an artificial life platform that encodes the topology of the network (and the weights of connections) in a fashion inspired by the encoding of gene regulatory networks in biological genomes. The number of computational nodes or connections is not limited in GReaNs, but here we limit the size of the networks to analyze the functioning of the networks and the effect of network size on the evolvability of robustness to noise. Our results show that even very small networks of spiking neurons can perform temporal pattern recognition in the presence of input noise.

Introduction

It is widely accepted that the brain employs temporal patterns of spikes to encode sensory input (for review, see Bialek et al., 1991; Gerstner et al., 1996; Laurent, 1996; Rieke et al., 1997; deCharms and Zador, 2000; Ahissar and Arieli, 2001; Huxter et al., 2003). Temporal coding forms the basis of sensory processing across different modalities, including hearing (Joris and Yin, 2007), vision (Thorpe et al., 1996), and olfaction (Isaacson, 2010). Nonetheless—in spite of the ubiquity of temporal coding in neuronal systems—the mechanisms that underlie the generation and recognition of temporal spike patterns are unclear.

It has been recognized that temporal pattern recognition can be performed by a system that contains an array of tuned delay lines and a coincidence detection mechanism (for example, Hopfield, 1995). Such a system, however, requires the existence of a developmental process or a learning algorithm that generates the appropriate delays, either by picking them from a spectrum of existing delays, or by adjusting them directly (Steuber and Willshaw, 1999; Steuber and De Schutter, 2002; Steuber and Willshaw, 2004).

In the present work, we evolve spiking neuron networks to recognize temporal input patterns without adjusting signal conduction delays. To evolve the networks, we modified the GReaNs (the name stands for Gene Regulatory evolving artificial Networks; Wróbel and Joachimczak, 2011) artificial life platform to allow for spiking nodes (Wróbel et al., 2012a). GReaNs has been used previously to evolve gene regulatory networks for tasks including controlling multicellular development in three dimensions (e.g., Joachimczak and Wróbel, 2008, 2012a,b), processing signals (Joachimczak and Wróbel, 2010b), and controlling animats (Joachimczak and Wróbel, 2010a; Wróbel et al., 2012b).

The next section briefly describes how the spiking neural networks (SNNs) are encoded in an evolving linear genome in GReaNs. We then follow with the description of the task for which the networks were evolved—recognition of a pattern of spikes, with varying levels of noise affecting spike times. We end by discussing the results and the way in which the networks perform the computation.

Evolving spiking neural networks encoded in a linear genome

The encoding of spiking neural networks (SNNs) in GReaNs is inspired by the encoding of gene regulatory networks in biological linear genomes (Wróbel et al., 2012a). Linear genomes are lists of genetic elements (Fig. 1), in our case each is a vector of 4 numbers that can mutate during evolution (type, sign, coordinate 1, coordinate 2). There are two main types of elements, which we call: P and G. P elements correspond to post-synaptic terminals (and are inspired by promoters in biological genomes). G elements correspond to pre-synaptic terminals (and are inspired by ggenes). A chain of P elements followed by a chain of G elements encodes a node in the network (one spiking neuron). Each node must have at least one G element followed by at least one G element.

Synaptic weights depend on the position in an abstract affinity space of points specified by elements' coordinates. If a *G* element and a *P* element specify points that overlap, the synaptic connection has maximum weight. The larger the distance, the lower the weight (using the inverse exponential function), provided the distance is below a prespecified threshold (to prevent full connectivity). The ele-

ments' sign defines if the synapse is excitatory (if the signs are the same) or inhibitory (if they are different).

In addition to *P* and *G* elements, *I* and *O* elements in the genome encode inputs and outputs, respectively, working as clamped pre- and post-synaptic terminals. The connectivity between *I* and *P* elements, and *G* and *O* elements is specified as described above for the connections between *G* and *P*. No other connections (or directions) are allowed. Importantly, the number of pre-synaptic and post-synaptic terminals is not limited in GReaNs (neither for the network, nor individual neurons). The number of neurons in the network is also not limited in the model (but we have limited—and varied—it in the experiments described in this paper).

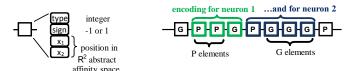


Figure 1: The structure of the genome in GReaNs. Each genetic element (left part) has integer value for type (*P*, *G*, *I*, or *O*), sign (1 or -1; the signs of two elements determine if a synapse is inhibitory or excitatory), and the coordinates (which determine the weights of the synapses). The genome (right part) is a chain of genetic elements.

Genetic algorithm

The genome is evolved with a genetic algorithm, in which genetic operators are applied on the level of the elements (here: with probability 0.0005—change of type, probability 0.0005—change of sign, with probability 0.02—sampling two random values from a Gaussian distribution centred at 0, and adding one value to each coordinate), and at the level of the whole genome (here: adding a chain of random elements, deleting a chain of elements, duplicating a chain elsewhere; each of these operators is considered at each element in the parent genome, with probability 0.1, and then the length of the chain is sampled from a logarithmic distribution with mean 10, and the site of duplication from a uniform distribution).

The genetic algorithm is initialized with a population with random genomes. In this paper, the population size was constant throughout all evolutionary runs (300 genomes). All elements in the initial populations had coordinates sampled from a Gaussian distribution, and all the initial genomes in each experiment encoded a network with five intraneurons, encoded by the chains of P and G elements with the length sampled from a Gaussian distribution (with mean = 1, and standard deviation = 1; if the length smaller than 1 was sampled, it was set to 1).

To calculate the fitness of the genomes (see below), the performance of the network encoded by each genome is tested. After selection (size two tournament), a new population is created, by applying the genetic operators described above to the best genomes, plus recombination (here: with probability 0.5 per genome, one-point cross-over with a random genome from the parent population, sampled uniformly). In the experiments described here, we used elitism—the five best genomes were copied (when new generations were formed from the old) without applying any genetic operators. Crossover and mutations were applied to 100 genomes, and only mutations—to the remaining 195 genomes. All the evolutionary runs in this paper were halted after 1000 generations.

Evolving networks of Leaky Integrate and Fire neurons for temporal pattern recognition

In this paper, we use Leaky Integrate and Fire (LIF) (Dayan and Abbott, 2005) neurons. GReaNs allows to use also Adaptive Exponential (Gerstner and Brette, 2005) neurons (Wróbel et al., 2012a), but they were not employed in the experiments described here.

LIF is one of the simplest and most commonly used spiking neural models. The rate of change of membrane potential is specified by the equation

$$\dot{V} = \frac{g_L(V_R - V) + g_E(E_{rev,E} - V) + g_I(E_{rev,I} - V)}{C}$$
(1)

where V is the membrane potential, V_R = -65.0 mV is the resting potential, g_L = 0.05 μ S is the leak conductance, g_E is the conductance of the excitatory synapses, g_I is the conductance of the inhibitory synapses, $E_{rev,E}$ = 0 mV is the reversal potential of the excitatory input, $E_{rev,I}$ = -70.0 mV is the reversal potential of inhibitory input, and C = 1 nF is the capacitance of the membrane.

When the postsynaptic potential of a neuron reaches the threshold voltage (V_{th}) , the neuron fires a spike and the value of the membrane potential is changed to the reset voltage (V_{reset}) .

In the GReaNs implementation of LIF neurons, the postsynaptic conductance increases by a small value proportional to the weight of the synapse when the pre-synaptic neuron (or input) spikes, and then the conductance decays exponentially:

$$\dot{g} = \frac{-g}{\tau} \tag{2}$$

where $\tau = 5.0$ ms, the decay time constant, is the same for both the excitatory and inhibitory synaptic conductance.

In the genetic algorithm, we used the following error function as the fitness measure (rewarding lower values):

$$f_{err} = 1 - \frac{min(S_{desired}, \alpha S_1) - \beta \sum_{i=2}^{6} S_i}{S_{desired}}$$
(3)

 S_i is the number of spikes generated by a network when it is connected to the pattern number i during the period between 250 and 1000 ms, $S_{desired}$ is the desired number of spikes to be generated by the network when it is connected with the correct pattern (here: 250), while α and β are constant fractions (here: $\alpha = 1$, and $\beta = 0.2$).

In other words, the task for which the networks are evolved is the recognition of the correct pattern, rewarded by the the first term, $min(S_{desired}, \alpha S_1)$, in the numerator. The correct pattern used in the experiments described here comprised three spikes, each spike arriving from one imput: one spike from input 1 at 50 ms, one from input 2 at 150 ms, and one from input 3 at 250 ms. We will call this pattern 1-2-3. Network activity in response to the other five patterns, with the same spike timings, but a different order (for example, 2-3-1, in which the first spike is from input 2), were penalized by the second term, $\beta \sum_{i=2}^6 S_i$. When evolving pattern recognition in the presence of

When evolving pattern recognition in the presence of noise in the input, temporal noise was added to spike times (by adding a temporal shift, sampled from the Gaussian distribution centered at zero, and with the standard deviation 10, 20, or 30 ms). We performed 10 independent evolutionary runs for each setting (the size of the network, and the level of noise; 160 runs in total). During evolution, the networks were evaluated one time when they were evolved without noise (with all the patterns, six simulations in total), or 100 times when they were evolved with noise (600 simulations in total)—and the f_{err} value was calculated as the average of these 100 values. After the runs ended, the champions were tested in the same fashion, to obtain the performance of the best among the champions, the average (among 10) and the standard deviation.

Results

To investigate the effect of network size on performance, we studied the evolution of networks limiting the maximum number of neurons allowed. In GReaNs, there are no constraints on the size of the genome, and thus the size of the network (even though available computational resources may restrict the feasible network size in practice). In the experiments described here, we simply ignored the rest of the genome after decoding the required number of nodes (intraneurons, inputs, and outputs).

For the simplest network, with only one neuron (Fig. 2), there is only one possible topology, since the direct connection between inputs and output was not permitted. To detect the 1-2-3 pattern, this single neuron needs to have a strong excitatory connection to the output (so that the output will start spiking when there is a spike at 50 ms in input 1), and an excitatory recurrent connection from the intraneuron to itself so that it continues spiking. We could not, however, evolve networks with just one interneuron that would be robust to noise (Table 1)—for the reason that will become clear

after analyzing the behavior of robust networks with more interneurons.

The networks evolved with the maximum number of two interneurons can evolve robustness to noise (Table 2) and their functioning can be analyzed in detail. Our analysis of champion networks from 10 independent evolutionary runs with the temporal noise of 10 ms allows us to classify them into three categories:

Category 1 In 3 out of 10 champions, one interneuron (A) acts as a detector of a spike in input 2; once this interneuron starts firing at high frequency (because of a recurrent connection), the second interneuron (B) goes to a plateau state—the membrane potential is higher than the resting potential, but still sub-threshold—interneuron B will start spiking once a spike from input 3 arrives. Because of this plateau state, the activation of the interneuron B does not depend on the time between spikes in input 2 and input 3 (which allows for the robustness to noise). Input 1 inhibits both interneurons, so they remain silent with input patterns 2-3-1 or 2-1-3. With 1-2-3, interneuron B starts spiking at 350 ms (100 ms after the spike on input, at 250 ms), allowing for robustness to noise on input 1.

Category 2 In 5 out of 10 champions, one interneuron (A) starts spiking after the spike in input 1, and thanks to the positive feedback loop, it keeps firing, putting the other interneuron (B) in a plateau state. When the spike from input 2 arrives, interneuron A starts firing at a higher rate, putting interneuron B in a higher plateau state. Only a small stimulus—a spike from input 3—is now required so that interneuron B starts spiking, and thus also the output. Here also the plateau state allows for robustness to noise—the times between spikes in input 1 and 2, or 2 and 3 have no effect on the pattern recognition.

Category 3 In 2 champions out of 10, one interneuron (A) acts as a detector of the time interval between spikes in input 1 and 2. If this interval is higher than the network was evolved for, the pattern is not recognized. Otherwise, once interneuron A puts the second interneuron in the plateau state, any timing of the spike in input 3 is sufficient for correct pattern recognition. This means, however, that although the networks in category 3 have zero error when tested with 10 ms noise (for which they were evolved), they are robust to higher noise levels only on the timing of the spike in input 3—in contrast to the networks in the categories 1 and 2, which are robust to higher noise levels on the timing of spikes in all the three inputs.

Even though just two interneurons allow for robustness to noise, allowing for more increases the performance (Tables 3, 4). Although these networks are too large to allow for the analysis as detailed as above, their functioning depends, similarly, on recurrent feedback loops that put some neurons in a high firing rate, and others at sub-threshold plateaus, so

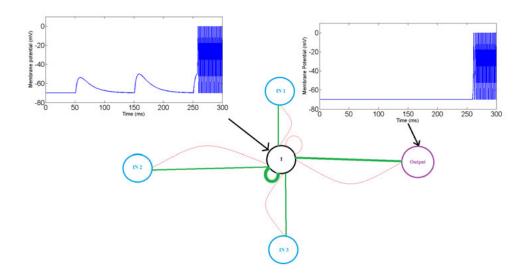


Figure 2: The final SNN with three inputs (cyan) each with only one spike at the times: 50 ms, 150 ms, and 250 ms, respectively, one interneuron (black), and one output (purple). The neurons are connected with both excitatory connections (green) and inhibitory connections (red). Line thickness indicates total synaptic weight.

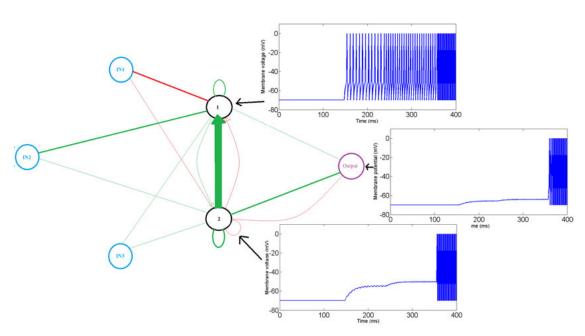


Figure 3: A champion SNN with two interneurons that belongs to category 1. The network has three inputs (cyan), which spike (in the correct pattern) at times 50 ms, 150 ms, and 250 ms, respectively; two interneurons (black), and one output (purple). The neurons are connected with both excitatory connections (green) and inhibitory connections (red). Line thickness indicates total synaptic weight.

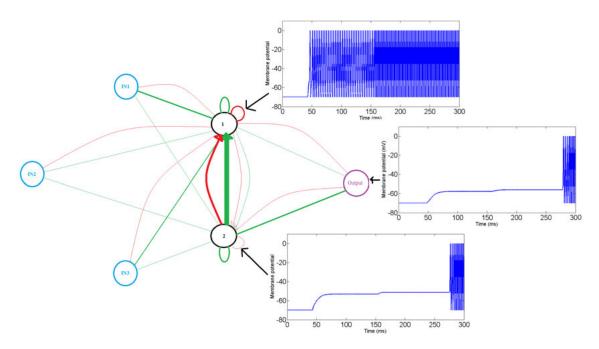


Figure 4: A champion SNN with two interneurons that belongs to category 2. The network has three inputs (cyan), which spike (in the correct pattern) at times 50 ms, 150 ms, and 250 ms, respectively; two interneurons (black), and one output (purple). The neurons are connected with both excitatory connections (green) and inhibitory connections (red). Line thickness indicates total synaptic weight.

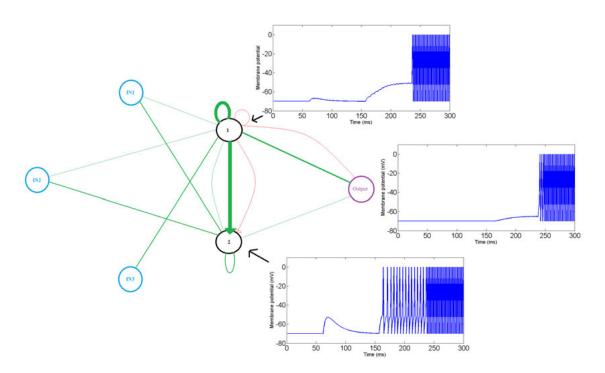


Figure 5: A champion SNN with two interneurons that belongs to category 3. The network has three inputs (cyan), which spike (in the correct pattern) at times 50 ms, 150 ms, and 250 ms, respectively; two interneurons (black), and one output (purple). The neurons are connected with both excitatory connections (green) and inhibitory connections (red). Line thickness indicates total synaptic weight.

Table 1: Evolution of robustness to noise in temporal pattern recognition in LIF networks with one intraneuron. The networks evolved with various levels of noise in the input (Gaussian noise added to spike times, centered at zero and with indicated standard deviation, sd) were tested with the same (or different) level of noise that (than) was used during evolution. The first value shows the performance of the best network in 10 runs, the values in brackets are averages \pm sd.

Testing	Temporal noise during evolution			
resung	sd=0 ms (no noise)	sd=10 ms	sd=20 ms	sd=30 ms
sd=0 ms (no noise)	$0(0.09 \pm 0.06)$	$0.05 (0.056 \pm 0.012)$	$0.05 (0.07 \pm 0.04)$	$0.049 (0.08 \pm 0.076)$
sd=10 ms	$0.017 (0.48 \pm 0.21)$	$0.198 (0.25 \pm 0.06)$	$0.2 (0.23 \pm 0.1)$	$0.196 (0.235 \pm 0.1)$
sd=20 ms	$0.042 (0.55 \pm 0.2)$	$0.29 (0.39 \pm 0.05)$	$0.22 (0.27 \pm 0.09)$	$0.216 (0.271 \pm 0.093)$
sd=30 ms	$0.11 \ (0.56 \pm 0.18)$	$0.4 (0.49 \pm 0.06)$	$0.25~(0.37\pm0.09)$	$0.229~(0.361\pm0.081)$

Table 2: Evolution of robustness to noise in temporal pattern recognition in LIF networks with two intraneurons. The networks evolved with various levels of noise in the input (Gaussian noise added to spike times, centered at zero and with indicated standard deviation, sd) were tested with the same (or different) level of noise that (than) was used during evolution. The first value shows the performance of the best network in 10 runs, the values in brackets are averages \pm sd.

Testing	Temporal noise	during evolution		
resung	sd=0 ms (no noise)	sd=10 ms	sd=20 ms	sd=30 ms
sd=0 ms (no noise)	$0(0.026 \pm 0.052)$	$0(0.04 \pm 0.07)$	$0.05 (0.18 \pm 0.2)$	$0(0.08 \pm 0.12)$
sd=10 ms	$0(0.36 \pm 0.22)$	$0(0.12 \pm 0.1)$	$0(0.18 \pm 0.066)$	$0(0.145 \pm 0.09)$
sd=20 ms	$0.04 (0.41 \pm 0.21)$	$0(0.21 \pm 0.16)$	$0.01 (0.2 \pm 0.1)$	$0(0.17 \pm 0.11)$
sd=30 ms	$0.08 (0.47 \pm 0.19)$	$0.05 (0.31 \pm 0.17)$	$0.06 \ (0.28 \pm 0.11)$	$0.07 \ (0.27 \pm 0.11)$

Table 3: Evolution of robustness to noise in temporal pattern recognition in LIF networks with five intraneurons. The networks evolved with various levels of noise in the input (Gaussian noise added to spike times, centered at zero and with indicated standard deviation, sd) were tested with the same (or different) level of noise that (than) was used during evolution. The first value shows the performance of the best network in 10 runs, the values in brackets are averages \pm sd.

Testing	Temporal noise during evolution			
resung	sd=0 ms (no noise)	sd=10 ms	sd=20 ms	sd=30 ms
sd=0 ms (no noise)	$0 (0 \pm 0)$	$0(0.06 \pm 0.13)$	$0.05 (0.09 \pm 0.11)$	$0.05 (0.06 \pm 0.02)$
sd=10 ms	$0(0.4 \pm 0.25)$	$0(0.07 \pm 0.09)$	$0.006 (0.18 \pm 0.06)$	$0.03 \ (0.18 \pm 0.05)$
sd=20 ms	$0(0.42 \pm 0.25)$	$0(0.13 \pm 0.12)$	$0.03 (0.23 \pm 0.07)$	$0.07 (0.2 \pm 0.05)$
sd=30 ms	$0.04~(0.5\pm0.23)$	$0.054 (0.24 \pm 0.15)$	$0.07 (0.33 \pm 0.01)$	$0.08 (0.29 \pm 0.08)$

Table 4: Evolution of robustness to noise in temporal pattern recognition in LIF networks with 10 intraneurons. The networks evolved with various levels of noise in the input (Gaussian noise added to spike times, centered at zero and with indicated standard deviation, sd) were tested with the same (or different) level of noise that (than) was used during evolution. The first value shows the performance of the best network in 10 runs, the values in brackets are averages \pm sd.

Testing	Temporal noise during evolution			
resuing	sd=0 ms (no noise)	sd=10 ms	sd=20 ms	sd=30 ms
sd=0 ms (no noise)	$0 (0 \pm 0)$	$0(0.02 \pm 0.06)$	$0(0.06 \pm 0.08)$	$0(0.16 \pm 0.08)$
sd=10 ms	$0.03 (0.4 \pm 0.29)$	$0(0.02 \pm 0.06)$	$0(0.05 \pm 0.08)$	$0(0.165 \pm 0.08)$
sd=20 ms	$0.14 (0.46 \pm 0.25)$	$0(0.04 \pm 0.08)$	$0(0.08 \pm 0.1)$	$0.01 (0.18 \pm 0.08)$
sd=30 ms	$0.4 (0.54 \pm 0.2)$	$0.07 (0.14 \pm 0.01)$	$0.02 (0.17 \pm 0.1)$	$0.05 \ (0.24 \pm 0.09)$

that they start firing when a spike from the next input arrives in the correct order.

Discussion

We can conclude from the analysis of the behavior of the SNNs with various numbers of interneurons that the feedback loops play a crucial role in transferring the network from one plateau state to another, until it becomes active. We can now also understand why the networks with just one interneuron cannot reach 0 error when noise is present during the evolution.

Importantly, once robustness evolved, the networks remained robust to higher and lower noise levels in the input than they experienced during evolution (Tables 2, 3, 4). This is in contrast to the results of the experiments on the robustness of multicellular development to noise on the activity of non-spiking nodes in the artificial gene regulatory networks evolved with GReaNs (Joachimczak and Wróbel, 2012a). In future work we plan to evolve robustness to noise introduced to the membrane potential of interneurons and/or the delays in the connections between them. Another interesting issue for further research is the relation between the evolvability and robustness to noise. Evolvability requires that mutations result in small phenotypic changes—a property related to robustness to noise, and to graceful degradation in the presence of damage (see e.g., Voigt et al., 2005).

The ability to recognize temporal input patterns is a characteristic that is shared by biological organisms at all levels of complexity. Temporal patterns of sensory inputs range from spatio-temporal gradients of nutrients in bacteria and protozoa to speech recognition in humans and echolocation in bats. In particular, vertebrate brains are believed to perform a large number of diverse temporal pattern recognition tasks (for review, see Bialek et al., 1991; Gerstner et al., 1996; Laurent, 1996; Thorpe et al., 1996; Rieke et al., 1997; deCharms and Zador, 2000; Ahissar and Arieli, 2001; Huxter et al., 2003; Joris and Yin, 2007; Isaacson, 2010).

Often, neural systems that perform temporal pattern recognition are thought to combine a conincidence detection mechanism with the ability to generate arrays of finely tuned time delays (Hopfield, 1995; Steuber and Willshaw, 1999; Steuber and De Schutter, 2002; Steuber and Willshaw, 2004; Steuber et al., 2006; Maex and Steuber, 2009). Animals are able, however, to recognize temporal patterns in the presence of large amounts of noise, and it is unclear how robust an array of finely tuned delay lines would be against noise in the temporal inputs patterns.

In the present work, we have investigated potential mechanisms of noise-robust temporal pattern recognition by evolving spiking neural networks for pattern recognition tasks, both in the presence and absence of noise. Our main result is that a spiking neural network can develop as a temporal pattern recognition device without being able to adjust signal propagation delays. Even without providing delay

lines, the temporal pattern recognition in the evolved networks is based on coincidence detection. But it is a particular form of coincidence detection—one that involves the sequential activation of a small number of neurons in the network. Moreover, positive feedback loops are the essential elements in these networks that provide robustness against temporal noise. Our results show that very small networks of spiking neurons can be able to perform surprisingly complex computational tasks in a noisy environment.

Acknowledgements

This work was supported by the Polish National Science Center (project EvoSN, UMO-2013/08/M/ST6/00922). AA is supported by the Foundation for Polish Science, cofinanced by EU Regional Development Fund (Innovative Economy Operational Programme 2007-2013). Computational resources were provided by the Tri-City Academic Computer Centre (TASK), and the Interdisciplinary Centre for Molecular and Mathematical Modeling (ICM, University of Warsaw; project G33-8). We are grateful to Michał Joachimczak for his help in debugging GReaNs.

References

- Ahissar, E. and Arieli, A. (2001). Figuring space by time. *Neuron*, 32:185–201.
- Bialek, W., Rieke, F., de Ruyter van Steveninck, R. R., and Warland, D. (1991). Reading a neural code. *Science*, 252:1854–1857
- Dayan, P. and Abbott, L. (2005). Theoretical Neuroscience. MIT Press.
- deCharms, R. C. and Zador, A. (2000). Neural representation and the cortical code. *Neuroscience*, 23:613–647.
- Gerstner, W. and Brette, R. (2005). Adaptive exponential integrateand-fire model as an effective description of neuronal activity. *Neurophysiology*, 94:3637–3642.
- Gerstner, W., Kempter, R., van Hemmen, J. L., and Wagner, H. (1996). A neuronal learning rule for sub-millisecond temporal coding. *Nature*, 383:76–78.
- Hopfield, J. (1995). Pattern recognition computation using action potential timing for stimulus representation. *Nature*, 376:33–36.
- Huxter, J., Burgess, N., and O'Keefe, J. (2003). Independent rate and temporal coding in hippocampal pyramidal cells. *Nature*, 425:828–832.
- Isaacson, J. (2010). Odor representations in mammalian cortical circuits. *Current Opinion in Neurobiology*, 20:328–331.
- Joachimczak, M. and Wróbel, B. (2008). Evo-devo in silico: a model of a gene network regulating multicellular development in 3D space with artificial physics. In Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems, pages 297–304. MIT Press.

- Joachimczak, M. and Wróbel, B. (2010a). Evolving gene regulatory networks for real time control of foraging behaviours. In Artificial Life XII: Proceedings of the Twelfth International Conference on the Simulation and Synthesis of Living Systems, pages 348–355. MIT Press.
- Joachimczak, M. and Wróbel, B. (2010b). Processing signals with evolving artificial gene regulatory networks. In *Artificial Life XII: Proceedings of the Twelfth International Conference on the Simulation and Synthesis of Living Systems*, pages 203–210. MIT Press.
- Joachimczak, M. and Wróbel, B. (2012a). Evolution of robustness to damage in artificial 3-dimensional development. *Biosystems*, 109:498–505.
- Joachimczak, M. and Wróbel, B. (2012b). Open ended evolution of 3d multicellular development controlled by gene regulatory networks. In *Artificial Life XIII: Proceedings of the 13th International Conference on the Simulation and Synthesis of Living Systems*, pages 67–74. MIT Press.
- Joris, P. and Yin, T. C. T. (2007). A matter of time: Internal delays in binaural processing. *Trends in Neurosciences*, 30:70–78.
- Laurent, G. (1996). Dynamical representation of odors by oscillating and evolving neural assemblies. *Trends in Neurosciences*, 19:489–496.
- Maex, R. and Steuber, V. (2009). The first second: Models of short-term memory traces in the brain. *Neural Networks*, 22:1105–1112.
- Rieke, F., Warland, D., de Ruyter van Steveninck, R. R., and Bialek, W. (1997). Spikes: Exploring the Neural Code. MIT Press.
- Steuber, V. and De Schutter, E. (2002). Rank order decoding of temporal parallel fibre input patterns in a complex purkinje cell model. *Neurocomputing*, 44:183–188.
- Steuber, V. and Willshaw, D. (1999). Adaptive leaky integrator models of cerebellar purkinje cells can learn the clustering of temporal patterns. *Neurocomputing*, 26:271–276.
- Steuber, V. and Willshaw, D. (2004). A biophysical model of synaptic delay learning and temporal pattern recognition in a cerebellar purkinje cell. *Journal of Computational Neuroscience*, 17:149–164.
- Steuber, V., Willshaw, D., and van Ooyen, A. (2006). Generation of time delays: simplified models of intracellular signalling in cerebellar purkinje cells. *Computation in Neural Systems*, 17:173–191.
- Thorpe, S., Fize, D., and Marlot, C. (1996). Speed of processing in the human visual system. *Nature*, 381:520–522.
- Voigt, C. A., Mayo, S. L., Wang, Z.-G., and Arnold, F. (2005). Directing the evolvable: Utilizing robustness in evolution. In Jen, E., editor, *Robust Design: Repertoire of Biological, Ecological, and Engineering Case Studies*, pages 105–134. Oxford University Press.
- Wróbel, B., Abdelmotaleb, A., and Joachimczak, M. (2012a). Evolving networks processing signals with a mixed paradigm, inspired by gene regulatory networks and spiking neurons. In *Proceedings of the 7th International Conference*

- on Bio-Inspired Models of Network, Information, and Computing Systems (BIONETICS), page in press. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering.
- Wróbel, B. and Joachimczak, M. (2011). Using GReaNs (genetic regulatory evolving artificial networks) for 3-dimensional asymmetrical pattern formation and morphogenesis. In *Proceedings of DevLeaNN: A Workshop on Development and Learning in Artificial Neural Networks*, pages 26–28, Paris, France.
- Wróbel, B., Joachimczak, M., Montebelli, A., and Lowe, R. (2012b). The search for beauty: Evolution of minimal cognition in an animat controlled by a gene regulatory network and powered by a metabolic system. In From Animals to Animats 12: The 12th International Conference on the Simulation of Adaptive Behavior (SAB 2012), volume 7426 of Lecture Notes in Computer Science, pages 198–208. Springer.

Improving Robot Behavior Optimization by Combining User Preferences

Anton Bernatskiy¹, Gregory S. Hornby^{2,3}, Josh C. Bongard¹

 University of Vermont, Burlington, VT 05401
 University of California Santa Cruz, Santa Cruz, CA 95064
 NASA Ames Research Center, Mountain View, CA 94035 abernats@uvm.edu

Abstract

Recently it has been demonstrated that collaboration between automated algorithms and human users can be especially effective in robot behavior optimization tasks. In particular, we recently introduced a Fitness-based Search with Preferencebased Policy Learning (FS-PPL) approach, in which the algorithm models the user based on her preferences and then uses the model, along with the fitness function, to guide search. However, so far only interaction between a single human user and an evolutionary algorithm was considered. If multiple users contribute preferences, the algorithm must determine whether to model them separately or jointly. In this paper we describe an algorithm in which one evolutionary algorithm interacts with two users and determines the best way to model them automatically. We test the algorithm with automated substitutes for human users and show that it performs better for two users working together than for the same users working separately, thus demonstrating the potential for crowdsourcing robot behavior optimization.

Introduction

Historically, interactive evolutionary algorithms are typically used to solve search problems in which automatic evaluation of a solution candidate is impractical for some reason – for example, artistic tasks. In this case the duty of solution evaluation is fully entrusted to the user. Many successful algorithms were designed using this approach, including those which allow multiple users to collaborate on the same problem (Secretan et al. (2008); Szumlanski et al. (2006); Kuzma et al. (2009)). Much less is known, however, about the algorithms which distribute the burden of solution candidates evaluation between the users and the computer.

In this work we employ this latter approach to address an important issue arising in traditional fitness-based evolutionary algorithms – namely, the phenomenon of premature convergence, i.e. convergence to a local optimum with a large basin of attraction rather than to the global optimum with a much narrower basin. One approach used to combat this problem is to use multiple objectives instead of just a single fitness value to evaluate solutions. Some objectives

shown to be effective are age (Schmidt and Lipson (2011)) and novelty (Mouret (2011)). However, depending on a task, even multiobjective algorithms can become trapped on local optima.

For some tasks this problem can be greatly reduced by adding human preference as an optimization objective. This is particularly true for robot behavior optimization, because humans have good intuition about legged locomotion and are able to visually determine that search has become trapped on a local optimum (Bongard et al. (2012)). The major problem with these methods, however, is the quantity of preferences required from the user, which is often so demanding that it makes the algorithm too labor-intensive to be practical.

This problem can be approached in several ways. One way is to use a machine learning algorithm to build a model of the user and then use the model to supply preferences on the human user's behalf as behavior optimization continues (Takagi (2001); Schmidt and Lipson (2006); Akrour et al. (2011); Bongard and Hornby (2013)). In (Bongard and Hornby (2013)) we investigated the efficiency of this approach in a robot behavior optimization task with a deceptive fitness landscape. Using an algorithm based on Age-Fitness Pareto Optimization (AFPO) (Schmidt and Lipson (2011)) with an additional user preference objective and a neural network-based user model, we showed that a user model and fitness function together can guide the search to convergence more rapidly (in terms of wall-clock time) than either of them on its own.

Another way to cope with the labor intensity of interactive evolution is to utilize evaluations coming from multiple users. This approach has been investigated theoretically to some extent (Szumlanski et al. (2006)) and successfully applied to artistic tasks (Secretan et al. (2008); Kuzma et al. (2009)).

Our hypothesis is that it is possible to make the optimization of robot behavior faster by collecting evaluations simultaneously generated by multiple users into one common evolutionary algorithm. Consider an algorithm which attempts to learn preferences supplied by multiple users based on

¹Fitness landscapes with such optima are said to be *deceptive*.

their evaluations. If n users simultaneously indicate preferences and if their preferences agree, then the machine learning algorithm can train on these preferences as if they were indicated by a single user. Therefore, it will have up to n times more training data, which will allow it to build an accurate user model faster.

If user preferences disagree, the algorithm will have to model users separately using their respective preference sets. In this case the speed of learning of each user model is reduced back to the level of the single user case, and additional computational costs associated with training multiple user models can impact the performance of the behavior optimization method (see the Experiments section). However, disagreement in users' preferences is likely to indicate that more than one global optimum – or several similar (in terms of fitness) local optima – have been intuited by the users and are present in the fitness landscape. In the latter case it is possible to exploit the disagreement to explore both of the user-favored optima, evaluate them and determine if one of the user-favored optima is better than the other in terms of fitness.

To test these suppositions we have developed an interactive, user-modeling algorithm which can simultaneously accept preferences from one or two users. We measure its performance with two users working together and compare it to the combined performance of two users working separately, each with her own evolutionary algorithm and user model.

Test Problem

We use the test problem from (Bongard and Hornby (2013)). The goal is to navigate a simple quadrupedal robot around the wall to a target object on the far side (Fig. 1a). The robot is composed of a square plate and four rigid vertical legs, each attached to the plate by an actuated joint with one degree of freedom (Fig. 1b).

Each body part has one light sensor and one touch sensor. Signals from the photosensors are real values from [0,1] varying linearly depending on their euclidean distance from the light source. 2 . Touch sensors produce 1 if the body part touches the ground or collides with the wall and -1 otherwise. Additionally, the robot is equipped with a compass sensor which gives the current robot's orientation relative to the Y axis, normalized to be in [0,1].

The robot is controlled by a feedforward neural network without hidden nodes. A total of 11 sensors connect to four actuators, which yields a total of 44 synaptic weights. Hereafter we will refer to a particular set of synaptic weights as a controller.

Methods

The algorithm uses a client-server computational architecture. The client here is an interactive program which takes

a pair of controllers as input, simulates³ two copies of the robot with controllers from the pair and shows the resulting behaviors to the user (Fig. 2). The user is forced to prefer one – she cannot skip a pair. After the preference is provided, the client sends it to the server.

The server performs the following functions:

- it supplies controllers to and receives preferences from multiple clients via asynchronous communication;
- it optimizes the robot's behavior with an evolutionary algorithm;
- it generates the controller pairs to be evaluated by users and maintains the users' preference tables;
- it trains the user models based on users' preferences;
- it employs predictions from the user models along with the fitness function to guide the evolutionary algorithm.

A *user model* is defined as a mapping from a pair of robot behaviors to a prediction of the user's preference for this pair. The mapping is learned by an artificial neural network⁴ with a hidden layer using backpropagation. For details, see the User Models section below.

If only one user has supplied preferences so far, only one user model is maintained. If two users supply preferences, the program must find an optimal way to utilize these. For this purpose our program maintains three separate user models – one *individual model* for each user and one *collective model*, which is trained on the combined preferences of both users. For details see the Coordinated Score Generation section below.

Evolutionary Algorithm

For robot behavior optimization the server uses Age-Fitness Pareto Optimization (Schmidt and Lipson (2011)), an evolutionary algorithm with two explicit objectives – fitness and age. In all experiments described below the algorithm starts with a population of 30 controllers, initialized with random synaptic weights in [-1,1]. The server simulates controllers sequentially and records the full time series of the resulting sensor values. When all controllers in the population have been simulated, the algorithm calculates their fitness values and constructs the Pareto front, taking the time controllers have spent in the population – their age – into account. The next generation is composed of

• one new, completely random controller,

²The sensors saturate to 0 for distances about 5 times greater than the maximum distance any robot traveled in our experiments.

³All physics simulations use Open Dynamics Engine, http://www.ode.org.

⁴This network is not to be confused with the robot's controller (see the Test Problem section), which is another artificial neural network employed in the program. Unlike the one described here that one has no hidden neurons.

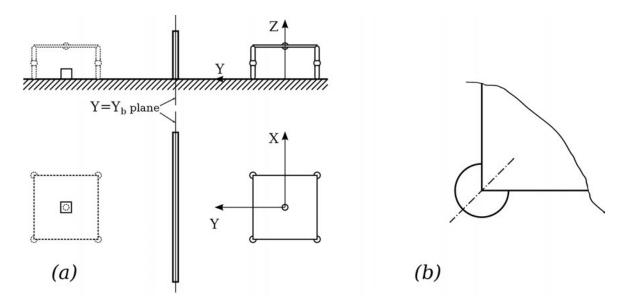


Figure 1: Test problem. (a) Side and top views of the robot and its environment at the beginning of the simulation. The small square to the left denotes the light source; spheres on the robot's body are light sensors. The target position that the robot should reach is depicted with dotted lines. Y_b denotes the Y coordinate of the barrier. (b) Joint between the robot's main body (square plate) and a limb, top view. The dotted line denotes the axis of rotation. The angle of the limb's rotation relative to its default position (as in (a)) can take values in $[-45^{\circ}, 45^{\circ}]$. A video of the robot with a successfully evolved controller can be viewed at http://youtu.be/ByDfAcDBsHI.

- nondominated controllers from the previous population and
- their mutated copies, in a quantity sufficient to restore the initial size of the population.

The fitness function is

$$f = f_u \sigma, \tag{1}$$

where f_u is the unscaled fitness (Bongard and Hornby (2013)):

$$f_u = \frac{1}{1 + \left(\sum_{i=1}^{5} \sum_{t=1}^{T} \|s_i^{(t)} - s_i^{(r)}\|\right) / 5T}.$$
 (2)

T=1000 here is the number of time steps during which behavior is simulated, $s_i^{(t)}$ is a value of ith light sensor at time step t, and $s_i^{(r)}$ is the value of the ith light sensor at the goal position (see Fig. 1a).

 σ is the *coordinated score*: a number in [0,1] which represents a combined prediction from all of the user models about how much the user (or users) would like this controller. In particular, σ near 1 indicates that at least one of the two user models tended to prefer this controller when it was presented multiple times, while a score near 0 indicates that the user models predict that both users will greatly dislike this controller. In the beginning of the program's operation, when no users' preferences have been provided yet,

it is equal to 0.5 for all controllers. For details on σ see Coordinated Score Generation section below.

In the current implementation, the second generation commences only after the first pair of controllers has been evaluated by a user. This ensures that the coordinated score σ affects evolution from the outset. However, in practice, this should have little impact on evolution, because the user models learn more slowly than the evolutionary algorithm improves the robot's behavior: it takes many before the user models' predictions deviate significantly from 0.5.

User Preference Gathering

After evaluating the first generation, the server ranks the controllers from the Pareto front by fitness and requests the evaluation of the four best controllers from the users. The first user must compare the first and the second controller, and the second user compares the third controller to the fourth. The program waits for either user to evaluate her pair and then enters the evolutionary loop of reproduction and selection (see the section above). The server never pauses to wait for any user action after the indication of this first preference

Every time the program evaluates all unscaled fitness values f_u of the controllers from the current generation, it checks whether any of its previous requests for user preferences were granted. If that is not the case, the program continues with the next iteration of the evolutionary loop. Otherwise, it stores the obtained preference into a table of

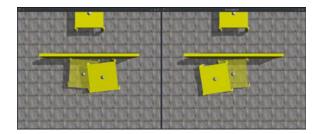


Figure 2: Screenshot of two clients running on the same computer. The user can select a behavior she likes by cycling through the robots. The selected robot is highlighted and the other one is made translucent. The preference is sent to the server as soon as the user confirms the selection. For example, in the left window the user is about to confirm her preference towards the highlighted robot to the right of the other contestant.

preferences, selects a new pair of controllers for user evaluation, sends it to the client and, if appropriate, retrains some user models on the expanded set of preferences.

All controllers sent to the client for user evaluation are stored, along with their respective sensor time series, in an archive. The obtained user preferences are stored in the preference table P, such that P[i,j]=1 if the ith controller of the archive was preferred to the jth, -1 if the jth controller was preferred to the ith, and 0 if the preference is neutral (in the current implementation that is possible only for P[i,i]) or not yet known (Bongard and Hornby (2013), Akrour et al. (2011)).

To accelerate the filling of the preference table we assume that user preferences are transitive. Consider a situation when the user has seen n controllers c_1, c_2, \cdots, c_n so far, and for every i < j she preferred c_j to c_i . The program assumes then that if a new controller c' is preferred over c_j for some $j \leq n$, then all controllers c_i (for which $i \leq j$) are assumed to not be preferred over c'. Similarly, if c' is not preferred over c_j , then all controllers from the upper part of the ranking, c_i (with $i \geq j$) are assumed to be preferred over c'.

To determine how a new controller fares against controllers previously shown to a user, the program uses a version of binary search adapted for our purposes. First, for each controller already shown it produces a score: the number of times this particular controller was preferred to its peers minus the number of times it was not preferred. If a new controller c' is preferred over some previously shown controller c_i , then it is assumed that c' is preferred over all previously shown controllers with a score less than or equal to the score of c_i , and the corresponding entries of P[i,j] are stored. Similarly, if some previously shown controller c_i is preferred to the new one, the algorithm assumes that all controllers with the score higher than that of c_i are preferred to the new controller.

The old controllers are shown to the user (paired with the new controller) in the following order: the controller with the highest score is shown first, then the one with the lowest score and then – repeatedly – the closest one to the middle of the current interval of possible values of the score for the new controller. The algorithm terminates when all of the relationships between the new controller and the previously known ones are established. In the worst case this happens after the user has indicated $2 + \log_2 n$ preferences; in the best case one preference is sufficient.

When the binary search described above terminates, two events occur. First, a new controller is selected among the current evolutionary population to be evaluated by the user. In the experiments described here, the algorithm selected the most fit controller among those which have not been seen by any user yet. The server sends the pair, as dictated by the first step of the bisection algorithm described above, to the

Second, two user models are retrained: the individual model corresponding to the newly gathered preference's author and the collective user model. The models are trained on the fully evaluated subsets of users' archives, i.e. on those subsets for which the preference is known for each pair of controllers in the subset. The individual model is retrained on the preference table of the user who indicated the last preference; the collective model uses the tables of both users.

This process of robot behavior optimization, preference gathering, and user modeling is repeated indefinitely, or until the server process is terminated.

Note that with the pair selection strategy described above a user never gets to evaluate a controller which has already been seen by her peer. The motivation for this is twofold. First, this approach maximizes the diversity of controllers available to the collective user model, which in turn maximizes its potential for accurate prediction. Second, it facilitates the detection of situations when the learned user models tend to overfit the user data (see the Discussion section).

User Models

A user model is a mapping between the robot's behavior and an assessment of its quality by the user. In this particular algorithm we employ a mapping which takes as input two robot behaviors compressed into feature vectors and maps them onto a value from [-1,1], approximating the record of the preference table P (see the User Preference Gathering section).

In all experiments described here we use the values from six sensors (five light sensors and one compass sensor) of the robot recorded at the middle (t=T/2) of the simulation as the feature vector (Bongard and Hornby (2013)). This kind of compressed representation simplifies the problem of learning the user model. Designing a general way in which such a vector can be generated to facilitate learning is

a nontrivial problem and it is not considered in this work.

The mapping is learned by an artificial neural network with 12 inputs – six for each feature vector of the two controllers which the model is supposed to compare. These neurons are connected to the only output of the network through a single hidden layer containing 12 neurons.

For convenience, the output neuron is trained to reproduce not the P[i, j] itself, but its linear transformation to [0, 1]:

$$target(i,j) \equiv \frac{P[i,j]+1}{2}. \tag{3}$$

The network is trained using error backpropagation (Rumelhart et al. (1986), Bongard and Hornby (2013)). The algorithm iterates through all entries of the preference table P[i,j] and backpropagates the network's errors associated with each entry once. If the network being trained is the collective user model, the same procedure is applied to the other user's preference table as well. Then it iterates through all of the entries again and compares the sign of the model's prediction to the sign of the original entry. If the signs coincide for all entries, the network is considered to be successfully trained. Otherwise, the procedure is repeated, but no more than $10^4(m/2-n)$ times, where m is the total number of table entries and n is the total number of controllers. If this number is reached, the learning process is considered to have failed.

Depending on the outcome of the learning procedure, the algorithm assigns *model errors* to each generated user model as follows:

- 10 if the learning failed;
- 2 if the learning was attempted on one preference table and succeeded;
- 1 if the learning was attempted on two preference tables and succeeded.

This value is used to determine the optimal way to utilize the three user models. As we will see in the next section, the behavior of the algorithm we use to accomplish that does not depend on the particular values we chose to represent the models errors, but rather on the relative position of these values on the real axis with respect to each other.

Coordinated Score Generation

To generate coordinated scores σ (see the Evolutionary Algorithm section above) for the newly-evolved controllers, the server starts by producing scores based on each one of the user models present (Bongard and Hornby (2013)).

To determine these scores for an evolutionary population of size 30, each user model fills a 30×30 table $\mathcal{P}[i,j]$ with its preference approximations (from [0,1]). The score is then

calculated as

$$\sigma_k(j) = \frac{1}{30} \sum_{i=1}^{30} \mathcal{P}_k[i, j],$$
 (4)

where $k \in \{0, 1, C\}$ is the index of the user model: 0 and 1 correspond to the first and second individual user models respectively and C corresponds to the collective model⁵.

Denoting the errors of the models (defined in the previous section) as ϵ_0 , ϵ_1 and ϵ_C , the coordinated score σ can be computed as follows:

- 1. If there is only one user model, use its score;
- 2. If $\epsilon_C < \epsilon_0$ and $\epsilon_C < \epsilon_1$, use σ_C ;
- 3. Otherwise, use $\max(\sigma_0, \sigma_1)$.

The first rule describes the trivial behavior the algorithm exhibits when only one user supplies preferences. The second corresponds to the condition under which the score from the collective user model should be used. With model error defined as we did in the previous section, this decision is always made when the backpropagation algorithm was able to train the collective user model successfully. The basis for this decision is the assumption that if it is possible to successfully train the collective user model on the data provided by two independent users, then these users are likely to be "allied", i.e., they are guiding the evolutionary search towards the same optimum.

The third rule describes the case when users are likely to have different opinions regarding which optimum is a global one. In that case the max function helps to retain controllers which are favored by one of the two users. This allows us to take both users' opinions into account and subject behaviors favored by each one of them to direct competition in the evolutionary algorithm.

We do not consider users who make errors or change their opinion over time in this work.

Experiments

To reduce the amount of effort required to test the algorithm and increase the experiments' reproducibility, we employed surrogate users in place of humans (Bongard and Hornby (2013)). A surrogate user is a version of the client program which emulates the behavior of a human user with particular preferences. In our experiments surrogate users preferred robots that attempt to circumnavigate around the right edge of the barrier, which is detected by measuring which one of the two controllers yields the largest X coordinate of the robot (Fig. 1) at the mid-point of the simulation (t=T/2)

⁵A similar metric was defined in the User Preference Gathering section to rank controllers by the degree to which a user likes or dislikes them. The value we generate here serves a similar purpose, but is computed using a different set of controllers.

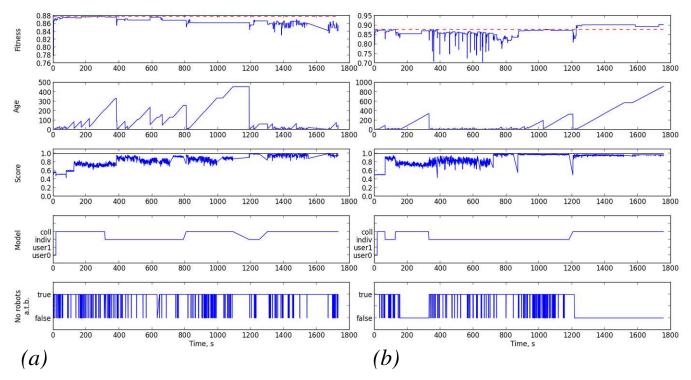


Figure 3: Time series for some parameters of the server over the course of typical allied simulations: (a) an unsuccessful simulation and (b) a successful simulation. The three topmost graphs represent the unscaled fitness f_u , age and the coordinated score σ of the current best controller in the evolutionary population (computed using the product $f_u\sigma$). The red dotted line in the fitness graph (top) shows a rough estimate of the maximal value of f_u which the robot not going around the barrier can have (0.88). The fourth graph from the top shows how the way in which the server modeled users changed over the course of the simulation: "coll", "u0" and "u1" indicate the usage of the collective user model and the individual models of the first and the second user, correspondingly; "indiv" corresponds to the two users being modeled separately. See the Coordinated Score Generation section for details. The graph at the bottom gives the logical value "No robots above the barrier": false if there are any controllers in the current population which make robot travel beyond the barrier (i.e., have $Y \geq Y_b$ at some point of the behavior simulation) and true otherwise.

(henceforth we will say that such a surrogate user *prefers* "rightmost" behavior). For emulating users with different strategies, we also made a version of the surrogate user who prefers behaviors with the lowest X coordinate at the same point of the evaluation period (i.e. a user who prefers "leftmost" behaviors).

Also, the surrogate user stopped supplying preferences and terminated the client if it encountered a controller which is able to guide the robot around the barrier, i.e., to have some points in its trajectory with Y greater than the coordinate Y_b where the barrier is located (see Fig. 1).

Results

In each run discussed below the server ran for 30 minutes of wall clock time. One or two clients controlled by the surrogate users were run on the same computer as parallel processes (Fig. 2). Once every 60 seconds the clients supplied preferences to the server.

Three types of simulation were performed:

- *single user simulations* with one surrogate user preferring "rightmost" behaviors;
- *allied simulations* with two surrogate users preferring "rightmost" behaviors;
- *opposing simulations* with one surrogate user preferring "rightmost" behaviors and one surrogate user preferring "leftmost" behaviors.

We considered a simulation to have succeeded if during the last 20 generations it had at least one controller in the server's evolutionary population which was able to guide the robot around the barrier (defined as above).

Figure 3 demonstrates how some parameters of the server change over the course of two typical allied simulations. Periodically, the age of the most fit controller stays constant for short time periods ("plateaus" on the age graphs). This occurs when the server is busy with model training for a significant portion of time and indicates the presence of a

				# of generations spent
Simulation	# of wins/	Rate	# of generations per run	using the collective model/
type	# of runs	of success	Average±std. deviation	Total # of generations
Single user	53/300	0.177	$(3.12 \pm 0.13) \times 10^2$	0/937443 (0%)
Allied users	86/300	0.287	$(2.59 \pm 0.28) \times 10^2$	538610/777250 (69%)
Opposing users	25/300	0.083	$(2.48 \pm 0.20) \times 10^2$	397220/744647 (53%)

Table 1: Experimental results

significant computational overhead related to training of the user models.

The graphs for the opposing simulations are very similar. Graphs for single user simulations differ from Figure 3 in two respects. First, there is no switching between usage of individual and collective user models to guide evolution: the algorithm has only one user model. Second, the amount of time the server spends training the user model is substantially lower.

We performed 300 runs of each of simulation type with the servers configured as described above. The results are presented in Table 1.

We used the one-tailed Z-test to compare the success rates (LeBlanc (2004)). The rate of success for allied simulations was found to be significantly higher than the success rate of the single user simulations ($p \leq 7 \times 10^{-4}$), despite the significantly lower ($p < 10^{-4}$ by the standard t-test) average number of evolutionary generations per run.

The success rate for the opposing simulations was found to be significantly lower ($p \le 7 \times 10^{-4}$) than the success rate of the single user simulations. The average number of generations per run is about the same as for the allied simulations, and is significantly less than the number of generations for the single user simulations ($p < 10^{-4}$).

The ratio between the number of generations which the server spent using the collective model and the total number of generations was found to be significantly higher ($p < 10^{-5}$) in the allied simulations than in the opposing simulations.

Out of all 25 opposing simulations which succeeded, at least 10 did so by taking the robot around the right side of the barrier and at least 9 used the left side of the barrier⁶.

Discussion

If a simulation involves two users, on average it iterates through fewer generations of the evolutionary algorithm than a simulation with only one user. This is explained as follows: in single user simulations the server maintains only one user model, which reduces the computational expense required for model training compared to the two user case in which three user models must be continually trained and

re-trained. This reduced computational burden is exploited by the evolutionary algorithm, which is now able to perform more generations.

The results also indicate that in allied simulations the program performs better than in the single user simulations. We explain this as follows: increased rate at which the common user model receives user preferences, coupled with the consistency of these preferences, causes the increase in the model's learning speed, yielding an accurate user model quicker (compared to the single user case). This result confirms our hypothesis that it is possible to accelerate robot behavior optimization by utilizing preferences from multiple users, despite the additional cost incurred by having to train individual and collective user models.

We hypothesize that the inferior performance of the program when it hosts opposing users is due to the three following factors:

- 1. When the coordinated score is generated as a maximum of scores by the individual user models, the evolutionary population is effectively divided into two subpopulations, each of which consists of controllers favored by the corresponding individual user model. This leads to a growth of the Pareto front and ultimately slows down search. We hypothesize that this problem may be remedied by utilizing an evolutionary algorithm which treats the Pareto front in a different way and/or employs a larger population.
- 2. In the experiments presented here, during a substantial fraction of generations (53%) opposing simulations employed the collective user model to guide search. The collective user model "successfully" learned a data set which has implicit internal inconsistencies. That is, the model must learn to take two similar inputs yet output two very different predictions: for example, the first user very much liked the first behavior, but the second user greatly disliked the second, similar behavior. This suggests that the model has overfit the data and its usage can negatively impact the algorithm's search ability.

Notice that if there were at least two controllers which both users has seen, it would become impossible to "successfully" train the collective user model. This can conceal the problem of overfitting. That's why, although such overlap would help the algorithm to recognize the situation when it is better to model users separately, it

⁶For the remaining six runs the wall was circumvented late into the simulation, which prevented the winning controller from being recorded.

is not allowed in the experiments reported here. This was one of the reasons why we decided to query the users on completely disjoint sets of controllers (see the User Preference Gathering.

This problem might be solved by using a different metric for the user model's learning efficiency.

3. The additional computational overhead mentioned above in the context of allied simulations already places this simulation at a computational disadvantage compared to the single user simulation.

Despite the general failure to accommodate opposing users, the algorithm still managed to solve the task in a substantial fraction of runs. These runs succeeded by discovering both user-favored optima, which indirectly confirms our second hypothesis about the possibility of finding and comparing multiple user-favored optima in the fitness landscape.

Conclusions

Our findings confirm that in robot behavior optimization tasks it is possible to increase the performance of fitness-based, user-assisted evolutionary algorithms by utilizing preferences from multiple users. This constitutes a step towards fitness-based, crowd-assisted algorithms which may potentially solve problems too deceptive to be solved by purely automated algorithms.

We demonstrated that employing more than one user can help solve robot behavior optimization tasks in at least two ways. First, if users approach the task with the same strategy, this approach allows the optimizer to recognize and employ the strategy more rapidly. Second, if the users employ different strategies, it is possible to find all optima recognized by the users and choose the best one among them.

However, the task of designing such algorithms is far from trivial. Here we would like to highlight some difficulties particular to search algorithms guided by multiple users, which employ user modeling. A good algorithm of this type must

- 1. be able to distinguish between different user strategies and model each appropriately;
- employ user modeling algorithms flexible enough to adapt to any or almost any benign user strategy, yet not overfit user input and thus retain good extrapolation properties; and
- 3. employ a search algorithm which is able to retain good performance while utilizing user models that change in number and quality.

Every one of these tasks constitutes a nontrivial design problem in its own right. However, we believe that all of these challenges can be addressed by a suitable combination of machine learning techniques. Possible future work may include utilizing clustering to solve the first subproblem listed above (pioneered in (Kuzma et al. (2009))), evolving user models of varying accuracy and complexity to address the second one and designing evolutionary algorithms with better scaling properties to handle the last one.

Acknowledgments

This work was supported by DARPA M3 grant W911NF-1-11-076.

References

- Akrour, R., Schoenauer, M., and Sebag, M. (2011). Preference-based policy learning. In *Machine Learning and Knowledge Discovery in Databases*, pages 12–27. Springer.
- Bongard, J. C., Beliveau, P., and Hornby, G. S. (2012). Avoiding local optima with interactive evolutionary robotics. In Proceeding of the fourteenth annual conference on Genetic and evolutionary computation conference, pages 1405–1406. ACM.
- Bongard, J. C. and Hornby, G. S. (2013). Combining fitness-based search and user modeling in evolutionary robotics. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*, pages 159–166. ACM.
- Kuzma, M., Jaksa, R., and Sincak, P. (2009). Clustering of users inputs in multi-user interactive evolutionary font design. In Applied Computational Intelligence and Informatics, 2009. SACI '09. 5th International Symposium on, pages 41–46.
- LeBlanc, D. C. (2004). Statistics: concepts and applications for science, volume 2. Jones & Bartlett Learning.
- Mouret, J.-B. (2011). Novelty-based multiobjectivization. In *New Horizons in Evolutionary Robotics*, pages 139–154. Springer.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.
- Schmidt, M. and Lipson, H. (2011). Age-fitness pareto optimization. In *Genetic Programming Theory and Practice VIII*, pages 129–146. Springer.
- Schmidt, M. D. and Lipson, H. (2006). Actively probing and modeling users in interactive coevolution. In *Proceeding of the eighth annual conference on Genetic and evolutionary computation conference*, pages 385–386. ACM.
- Secretan, J., Beato, N., D'Ambrosio, D. B., Rodriguez, A., Campbell, A., and Stanley, K. O. (2008). Picbreeder: evolving pictures collaboratively online. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1759–1768. ACM.
- Szumlanski, S. R., Wu, A. S., and Hughes, C. E. (2006). Conflict resolution and a framework for collaborative interactive evolution. In *AAAI*, pages 512–517. AAAI Press.
- Takagi, H. (2001). Interactive evolutionary computation: fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296.

Blueprint for Self-replicating and Self-assembling 3D Printers

Mads Buch^{1,2} and Steen Rasmussen^{1,3}
¹FLinT Center, University of Southern Denmark
²Aarhus University, Denmark
³Santa Fe Institute, New Mexico, USA
madspbuch@gmail.com and steen@sdu.dk

In the 1940s Stanislaw Ulam and John Von Neumann came up with the idea of cellular Automata (CAs), which Von Neumann later used to develop a universal constructor (Neumann, 1966). The universal constructor is a pattern implemented in a CA that consists of a two parts: The universal constructor and a tape. The universal constructor reads from the tape and produces what is encoded. A universal constructor may be encoded on the tape, in which case it produces a functional copy of its own the pattern. The term "universal" in its name refers to the fact, that it itself is a universal computer, and as such it can construct all possible (computable) patterns including patters of itself. The Artificial Life community has over the years developed a variety of simple self-replicating CA patterns, e.g. see (Sayama, 2000) and references therein. The CAs, however, are not well suited for programming of arbitrary patterns. Thus, they are unpractical as universal constructors. Biological systems are also clearly examples of constructors capable of making copies of themselves, although they are not universal. They cannot be programmed to construct arbitrary systems. Significant scientific activity has been devoted to construct simple physicochemical systems capable of reproducing themselves, see e.g. (Rasmussen et al., 2008).

Since Von Neumann's proved that mathematical machines "that can make anything" do exist, such machine implementations have been a favorite item in popular scientific fiction novels and -movies. In contrast, except for a few, early, original contributions on self-replicating machines, e.g. see (Zykov et al., 2005), it's only in recent years the scientific community again has gathered around this tantalizing concept and has revamped ideas about how to implement embodied universal constructors, see e.g. (Packard et al., 2010).

To explore blueprints for embodied universal constructors is the topic of this work.

We may view two main approaches for embodied, macroscopic, universal constructors. In the one extreme we have systems where most of the necessary functional components are assembled bottom up by the internal constructor, from the available raw materials. This approach approach is exemplified by biological systems. It is difficult, however, to

imagine how a macroscopic constructor, at least based on known technology, could do just that. In the other extreme, we have constructors that pick up already fully functional components and self-assemble these into new copy of themselves. A simple version of this approach is already implemented by Hod Lipson and colleagues (Zykov et al., 2005).

We choose a middle way for our constructor. Inspired by natural self-replicating systems, some components are reused and harvested from the environment while others are assembled internally bottom up. As the basis for embodiment we investigate extrusion based 3D printer technology and let the parent-printer print components which are simple, while other components either are assumed available from the environment. These components either have material- or technical constraints, which make them infeasible to print with current technology. Although the underlying code controlling a 3D extrusion printer in principle can encode any physical object and is a such universal, there are currently severe limitation on how general such physical objects can be. We will initially limit ourselves to self-replication of and self-assembly of component.

For the extrusion based 3D printer we need material with three types of electrical properties: An isolating material, a semiconducting material and a fully conducting material. With these types of material it would be possible to build computing components (control logic) and moving components (motors). Conducting- and none-conducting plastics already exist (Leigh et al., 2012), while semiconducting plastics are not year realized, but is intensively investigated. Thus, for the time being we may assume the semiconductor plastics are provided from the environment. Additionally we need a ceramic material to e.g. to build new extrusion heads. Ceramics can easily be extruded (as clay) and later be baked to further develop more heat resistant properties.

For the self-assembler we assume a system composed of a fixed frame with a grabbing arm, which has freedom to move, grab and join objects within a certain volume. Such a contraption poses significant engineering challenges, but since various industrial robots already in use perform all the necessary actions, it should be feasible to implement such a

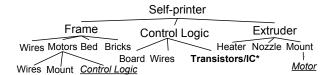


Figure 1: The component nodes and their dependencies (to the right) for the *Self-printer*. The leaves denotes atomic components either for producing of for retrieval from the environment. Emphasized leaves are replaced by the non-emphasized leave of the same name elsewhere.

self-assembler. It can be viewed as a generalized version of how Lipson's homogenous cubes are assembled.

A (graph) three is the natural way to express componentand process dependencies. We can therefore construct a tree expressing the components the printer needs to print, as well as a tree expressing the processes the self-assembler needs to perform, both to assemble itself and to assemble the printer.

The root node for the Self-printer is the main device itself, see Fig. 1. The Frame dependency refers to the construction that keeps all the parts together and allows for the movement of parts. The frame is printable and the implementation we use, is the Lego-printer (Møllerhøj, 2013) for the Brick-parts. Furthermore the frame depends on motors. Motors can be printed bottom up (Hawkins, 2013). The Bed (the surface that the childprinter will be printed on) should tolerate being heated, it should be flat, and it should allow the printing materials to stick. Ceramic materials may be used as a material for printing beds onto. Finally the frame needs wiring for the electronics as well as the motors. These are printable as strips of conducting plastic. Insulated wires are made by coating the conducting wires with insulator (just denoted wires in Fig. 1.). Control logic is constructed by means of PCB (Printed Circuit Board). PCBs are essentially nonconducting structures that holds wires and transistors, which are easily printed (Board in Fig.1). It is not (yet) possible to print *Transistors* as the semiconducting material is still a technological challenge. However, if all logic is made in only transistors, the logical parts would be very large. Thus it is not practical to implement all logic in transconducting- and a non-conductingtors. Therefor we utilize IC (Integrated Circuits), which are harvested from the environment, as they cannot either (yet) be made with extrusion technology. The last thing the self-printer is dependent on is the Extruder. Both the Heater and the extrusion Nozzle are printed in a ceramic material, conducting and nonconducting respectively. The Mount is the part that holds the hot-end and feeds it with plastic. The Motor can be printed, and the rest is trivial to print. The structure requires the hotend to include heat isolation. This could be done with a ceramic layer. We may interpret Frame, Control Logic and Extruder as functionally corresponding to Container, Information and Metabolism for minimal self-replicating physic-

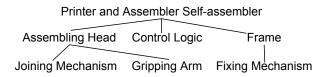


Figure 2: The dependencies in the self-assembly process. The assembling-process is specialized i.e. each individual component has a special procedure for building it (dependencies to the right).

ochemical systems (Rasmussen et al., 2008).

The *Printer and Assembler Self-assembler* node in Fig. 2 depends on the *Frame, Control Logic* and *Assembly Head*, which again have deeper dependencies as discussed previously. The build process is highly specialized and the components are set aside in the assembler until they are joined in their parent components. This macroscopic assembly process is very different from molecular, mainly entropy driven, self-assembly processes.

In this short presentation we focus our attention on addressing perhaps the most critical design challenges for self-replicating and self-assembling 3D printers given our laboratory experiences. We believe the component- and process dependency based design approach is appropriate for the problem at hand, and we believe the extrusion printer technology is promising as a simple first implementation technology for an early macroscopic self-replicating machine. An actual implementation of the proposed blueprint obviously raises significant practical problems. However, they should be addressable in a modern lab by a cross disciplinary team.

Hawkins, C. (2013). http://www.youtube.com/watch?v=pat63-8dlbs.

Leigh, S. J., Bradley, R. J., Purssell, C. P., Billson, D. R., and Hutchins, D. A. (2012). *PLoS ONE*, 7(11):e49365.

Møllerhøj, J. (2013). http://ing.dk/artikel/studerende-bygger-sinegen-3d-printer-i-lego-160721.

Neumann, J. V. (1966). *Theory of Self-Reproducing Automata*. University of Illinois Press, Champaign, IL, USA.

Packard, N., McCaskill, J., and Rasmussen, S. (2010). http://flint.sdu.dk/index.php?page=living-technology-personal-fabricator.

Rasmussen, S., Bedau, M. A., Chen, L., Deamer, D., Krakauer, D. C., Packard, N. H., and Stadler, P. F., editors (2008). *Protocells: bridging nonliving and living matter*. the MIT press, Cambridge (Mass.).

Sayama, H. (2000). Artificial Life, 5:343-365.

Zykov, V., Mytilinaios, E., Adams, B., and Lipson, H. (2005). *Nature*, 435(7038):163–164.

Evolution of Chemical Signals in ecological system evoked by the "cry-wolf" plants

Megumi Sakai and Yasuhiro Suzuki

School of Information and Sciences, Nagoya University ysuzuki@nagoya-u.jp

Abstract

We model the tritrophic system composed of plants, herbivores, and carnivores, where plants produce chemical signals when they suffer from feeding damage by herbivores; and this chemical, Herbivore Induced Plant Volatile (HIPV) attracts carnivores, thus plants can indirectly protect themselves from feeding damages caused by herbivores. Carnivores in this system are able to evaluate and learn its usefulness of the chemical signals, therefore plants do not emit the chemical signals until the population of herbivores becomes large enough for carnivores, where in the coupled tritrophic system, it has been confirmed that there are plants called "cry wolf plants" that emit chemical signals even if there are few herbivores. It has been pointed out that if there emerges cry wolf plants in this system, chemical signals may change in order to preserve the quality of information and keep on attracting carnivores. We model the tritrophic system including cry wolf plants, and we confirm that the chemical signal may change through simulations of the model. Further we show the chemical signal may not change when plants grow densely in the field.

1.Introduction

The tritrophic system composed of plants, herbivores, and carnivores has been investigated. In this system, when plants suffer from feeding damage by herbivores, plants produce Herbivore Induced Plant Volatile, HIPV; HIPV attracts carnivores to the plants and the attracted carnivores exclude the herbivores giving feeding damage, hence the plants are able to protect themselves from herbivores indirectly (for example, Takabayashi et al. 1988) and this system has been theoretically investigated (for example, Sabelis et al. 1988, 2002). In the tritrophic system, plants commonly produce HIPV in a dose-dependent manner: the more herbivores, the more volatiles are released.

The volatiles attract predatory carnivores and the amount of produced HIPV determines the probability of predator response. They show that seedlings of a cabbage variety (Brassica oleracea var. capitata, cv Shikidori) also show such a response to the density of cabbage white (Pieris rapae) larvae and attract more (naive) parasitoids (Cotesia glomerata) when there are more herbivores on the plant. However, Shiojiri et al. (Shiojiri et al. 2010) discovered when the plant is attacked by diamondback moth (Plutella xylostella) larvae, seedlings of the same variety (cv Shikidori) release volatiles of which the total amount is high and constant, and thus independence of the caterpillar density, and naive parasitoids (Cotesia vestalis) of diamondback moth larvae fail to discriminate herbivore-rich from herbivore-poor plants. In contrast, seedlings of another cabbage variety of B.

oleracea (var. acephala: kale) respond to the density of diamondback moth larvae in a dose-dependent manner and attract more parasitoids when there are more herbivores.

Assuming these responses of the cabbage cultivars reflect behaviors of some genotypes of wild plants at least, they provide arguments why the behavior of kale (B. oleracea var acephala) is best interpreted as an honest signaling strategy and that of cabbage cv Shikidori (B. oleracea var capitata) as a "cry wolf" signaling strategy, implying a conflict of interest between the plant and the enemies of its herbivores: the plant profits from the visits of the herbivore's enemies, but the latter would be better for the enemies to visit other plants with more herbivores. If so, evolutionary theory on alarm signaling predicts consequences of major interest to students of plant protection, tritrophic systems, and communications alike.

On the tritrophic system with cry wolf signal, Sabelis et al. (Sabelis et al. 2010) pointed out it may evoke co-evolution of signals; because once plants send the dominant honest signal, new opportunities arise for 'cry wolf' plants that mimic this signal, though they harbor no or few herbivores, if any. Thus, frequency-dependent selection will give rise to alternating waves of plants sending 'honest' or 'cry wolf' signals (Baalen, et al. 2003). This process is likely to increase the complexity of plant signals, and the theory on the evolution of cooperation has shown that the more complex the signal is, the more likely it evolves and perpetuates cooperative alliances (Traulsen et al. 2007). Thus they predict that chemical alarm 'languages' of plants change over generations and become complex due to frequency-dependent selection. Since the perception of odor blends seems not to be a simple sum of responses to individual components, but rather to be based on properties of the odor blend as a whole, small changes in the odor blend may allow the signal to be perceived as new and this may more easily give rise to new signals of plants by mutation and more easily for them to be selected.

The evolution of Herbivore-Induced Plant Volatiles (HIPV) shares common traits with the evolution of language (Baalen et al. 2003, Traulsen et al. 2007). In the evolution of language, a mutation is reflected by the change of symbols, and through natural selection, the mutated symbols become understandable and widely used, they called such a situation as the "tower of Babel." Likewise, in the evolution of HIPV, a mutation brings about a change in Herbivore-Induced Volatile Chemicals, and natural selection enables the mutated HIPV to become understandable and widely used in the ecosystem. A plant changes the response to its feeding damage through mutations.

2.Method

We model the tritrophic system on the two dimensional grid (torus) and each grid point has eight neighbors (the Moore neighbor), each grid spot has a plant or empty space; herbivores invade each plant and eat the plant. When a plant suffers from feeding damage, and if the amount of damaged biomass is large, then the plant starts to produce Herbivore Induced Plant Volatile, HIPV; produced HIPV diffuses on the space isotropically; carnivores search herbivores by tracking HIPV, if a carnivore can sense the gradient of HIPV, then it tracks the gradient, otherwise searches the HIPV randomly; each carnivore possesses "energy" and it is expensed through searching and if the carnivore find herbivores then it removes all herbivores in the plant and gains the energy in proportion to the number of herbivores in the plant; if a carnivore cannot find herbivores and spends all energy, it dies and is eliminated from the space.

2.1. The details of the model

Plants: In the simulation in this paper, all plants are in the same condition where they complete their growth for simplicity. The biomass of a plant is expressed as the given value, in the initial state it is set to 100 and decreases 2 because of feeding damage by a herbivore and when the biomass is below 98 because of feeding damage, it increases 2 by growth; the maximal value of biomass is 100 and it does not grow beyond 100; if the value of biomass is equal to zero, it dies and is eliminated from the space.

If there are herbivores or carnivores in the eliminated plant, they are moved to the nearest plants randomly; the use of the parameter indicates biomass which has the fixed maximal value and the maximal population of the plant is the total number of grids in the space, which means that the total size of biomass of plants is finite and limited. The growth of a plant is expressed as the increment of its biomass by 2 and when a plant grows and reaches 50% of the maximal biomass (in this paper, it is 500, when the plant has a descendant, then a descendant plant is placed in the neighbor of its parent plant and if there is no neighboring empty place, it is placed in the nearest empty place.

The initial biomass of the descendant is 100 and the descendant changes the type of HIPV according to the mutation rate of the type of HIPV; when a descendant plant does not mutate, it produces the same type of HIPV as its parent plant; the characteristics of plants such as "honest" or "cry wolf" can be only changed during the generation change. Each plant has the threshold value on generating HIPV (in this paper, the threshold value is 15); if the number of herbivores exceeds the threshold value, the plant starts generating

HIPV; HIPV is expressed as the non-negative natural number, the characteristics of each plant is "honest" or "cry wolf"; the threshold value of a honest plant is high and of a cry wolf plant, low. In the initial state of the simulation, every plant has the same HIPV: plants in a certain percentage to the total population change the type of HIPV. Herbivores; Herbivores invade a plant randomly; a plant is selected randomly from the field and a herbivore is put inside it; they grow up by reducing the biomass of the plant, and the

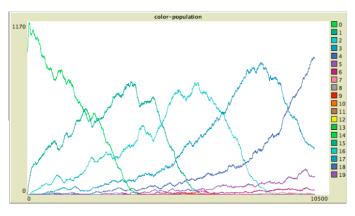
maximal number of herbivores in a plant is limited. When the population of herbivores reaches the maximum, herbivores move to the nearest plants that have not the maximal population of herbivores. If there are no neighboring plants they can move to, then they move to the nearest plant.

Carnivores; a carnivore moves 1 grid point by each step; it moves to the plant which produces the target HIPV. If there are no neighboring target plants, it searches the target HIPV and tracks the gradient of the HIPV concentration; when there is no neighboring target HIPV, it searches the same HIPV until it moves 5 steps among the neighboring 8 grids. If it is still unable to reach the target plant, it randomly selects a plant around the 8 grids. Each of carnivores has a memory and the memory affects the preference of HIPV, a carnivore evaluates the usefulness of HIPV and changes the preferences of HIPV; if a carnivore tracks a certain HIPV and it cannot obtain enough herbivores, then the carnivore devaluates the usefulness of HIPV and memorizes it.

The memory is realized as the First In First Out, FIFO queue of the type of HIPV which attracts a carnivore, where the number of memories of the HIPV type represents the preferences of HIPV and the maximal length of the queue expresses the characteristic of learning activity of a carnivore; if the maximal length is short, such a carnivore is likely to change its preference frequently, vice versa. For example, the memory of a carnivore is the queue of "1,1,1,2,3", where the each natural number represents the type of HIPV. If the length of the queue exceeds the maximal length, the oldest element is removed and the newest element is put in (such operation is called First In First Out manner). In this simulation, the maximal length of the queue is set to 100 and there is the same type of HIPV in the initial state. A carnivore selects the type of HIPV randomly from the queue and tries to track the HIPV in the space and if the usefulness of the HIPV is high, the selected type of HIPV is put in the queue and the oldest information is removed from the queue.

Plants		
Initial population:	65% of the total number of grid	
	points	
Generation rate of a	0.65 (This generation rate means	
new plant:	the generation rate of a new plant	
	but not a descendant).	
Feeding damage:	2	
Carnivores		
Generation rate of a	0.03 / step	
bug:	_	
Natural enemy		
Growth rate:	0.04 (the growth rate of the	
	enemy's population)	
Initial energy value:	500	
Traveling cost:	5 (a carnivore spends 5 energies at	
	every step).	
The maximum possible	5 (a carnivore can move up to 5	
travel distance:	steps at 1 travel.)	

Table 1. Parameters used in the simulations



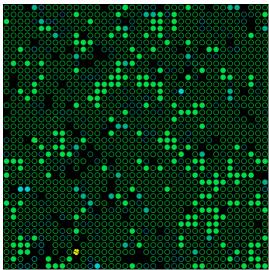


Figure 1. Sparsely distributed plants in the initial state: (above) the population dynamics of plants, where each line denotes the number of plants that generate the same type of HIPV; the vertical axis illustrates the number of plants and horizontal axis illustrates the step time. (below) The circles show the distribution of plants in the space at the 10,000 step times, the colored circle illustrates the HIPV produced by a cry wolf plant and the difference of the HIPV type is represented by a different color.

In the case when a carnivore does not find the target HIPV or cannot reach the patch of herbivores, the carnivore stops finding or tracking the HIPV and tries to track a different type of HIPV. A carnivore evaluates the usefulness of HIPV as the ratio of the number of herbivores in the patch to the total travel distance to the patch; the carnivore gains this ratio as "energy". The number of the type of HIPV to be put in the queue is the proportion to the result of the evaluation. If the evaluation of the HIPV is high, relatively large numbers of the type of HIPV are put in and the same numbers of information are removed from the memory.

3. Results

When there are no cry wolf plants, carnivores do not have to track a different type of HIPV other than the dominant type of HIPV, therefore carnivores do not visit plants that produce non-dominant HIPV, hence such plants go extinct (Figure 1). When there emerges cry wolf plants since the dominant HIPV is mimicked by emerged cry wolf plants, honest plants have to produce different types of HIPV in order to keep on attracting carnivores; such a change of HIPV is regarded as a "mutation" and if the changed HIPV can attract many carnivores, the plants that produce such HIPV are able to increase its population, which can be regarded as "selection"; it is noticed that we use the terms, "evolution", "mutation", and "selection" for analogical describing of the HIPV changes.

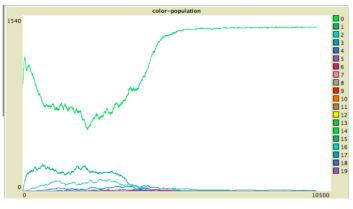
The evolution of HIPV is affected by the rate of HIPV change, we call the rate as mutation rate; when the mutation rate is high, there may emerge many types of HIPV, as we addressed in the previous section, the total biomass of plants is limited, so as the number of HIPV types is large, the population of plants that produce each type of HIPV becomes relatively small; hence in such a case, it is likely to be difficult for the carnivores to track and find the target HIPV, then plants are likely to go extinct; which is "the tower of Babel" addressed in the previous section.

3.1 Cry wolf plants can be honest plants

In the previous every simulation, we set plants sparsely on the space, next we only change the distribution of plants by placing them densely on the field, while all the other parameters and settings are the same.

The density of Cry Wolf plants expressed as the mutation rate between cry wolf plants and honest plants; in the preliminary simulations, we empirically found that if the mutation rate from honest plants to cry wolf plants is twice as much as the mutation rate from cry wolf plants to honest plants, cry wolf plants grow in clumps. Then, we use this parameter to make cry wolf plants grow in clumps, on the other hand, we double the mutation rate from cry wolf plants to honest plants to make honest plants grow in clumps. In this case, the HIPV does not evolve (Figure 2).

This is because the carnivores evaluate the HIPV produced by cry wolf plants as "honest signal"; otherwise if the carnivores evaluate such HIPV as cry wolf signal, they explore and change its preference and the evolution of HIPV should emerge; it was confirmed that the carnivore's preference of HIPV does not change through checking the time developments of the memory queue of the carnivore (Figure 2).



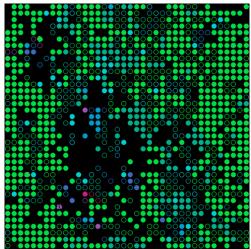


Figure 2. Densely distributed plants in the initial state: (left) the population dynamics of plants, where each line denotes the number of plants that generate the same type of HIPV; the vertical axis illustrates the number of plants and horizontal axis illustrates the step time. (Right) The circles show the distribution of plants in the space at the 10,000 step times, the colored circle illustrates the HIPV produced by a cry wolf plant and the difference of the HIPV type is represented by a different color.

4.Discussion

In this simulation, a carnivore evaluates the usefulness of HIPV as the ratio of the number of herbivores in the patch it reached to the total travel distance to the patch; hence comparing the travel distances between sparsely and densely distributed plants, when the distribution of plants is dense, the travel distances are likely to be shorter, thus, even if carnivores are attracted to the cry wolf producing HIPV and obtain few herbivores, the evaluation of the usefulness of HIPV may not be low, while if the distribution of cry wolf and honest plants is sparse, carnivores may devaluate the usefulness of such HIPV. On the contrary, if honest plants are densely distributed and cry wolf plants are sparsely distributed, the carnivore's travel distance to the cry wolf plants increases, then the evaluation of the HIPV by the

carnivore is low, and a new HIPV produced only by honest plants will be dominant and the HIPV will develop. However if cry wolf plants are sparsely distributed among densely distributed honest plants, the carnivore's travel distance is short, and then we consider it is unlikely that the new HIPV will be dominant.

As previous studies showed, when the spatial distribution of Cry wolf plants and honest plants is homogeneous, HIPV may become diverse because of the emergence of Cry wolf plants. However, for example in a field, when one type of plants grows in clumps, HIPV may not become diverse even though there are Cry wolf plants as this study showed.

Acknowledgements

This work was supported by the JSPS Core-to-Core Program (No.20004), JSPS KAKENHI Grant Numbers 23300317 and 24520106 and the Grant-in-Aid for Scientific Research on Innovative Areas Grand Number 2404002.

References

- van Baalen M., Vincent A., Jansen A. Vencent. (2003), Common language or Tower of Babel? On the evolution of signals and their meanings, *Proc. R. Soc. Lond.* B 270, 69-76.
- Shiojiri K., Ozawa R., Kugimiya S., Uefune M, van Wijk M., Sabelis M.W., Takabayashi J. (2010), Herbivore-specific, density-dependent induction of plant volatiles: honest or "cry wolf" signals? *PLoS One.* 2010 Aug 17; 5(8):e12161. MI. A. K.
- Sabelis M.W., A. Janssen A.. Takabayashi J., (2011), Can plants evolve stable alliances with the enemies' enemies?, Journal of Plant Interactions, Vol. 6, Nos. 2-3, June September71-75
- Sabelis MW, van Baalen M, Pels B, Egas M, Janssen A (2002). Evolution of Exploitation and Defense inTritrophic Interactions. In: Adaptive Dynamics of Infectious Diseases: In Pursuit of Virulence Management, eds. Dieckmann U, Metz JAJ, Sabelis MW & Sigmund K, pp. 297–321. Cambridge University Press. for a polymorphic ESS of synomone production in plants. Oikos 53:247–252
- Sabelis MW & De Jong MCM (1988). Should all plants recruit bodyguards? Conditions for a polymorphic ESS of synomone production in plants. Oikos 53:247–252
- Suzuki Y. (2009), Behaviors of Chemical Reactions with Small Number of Molecules. Lecture Notes in Computer Science, 5777, 349-401, Springer Verlag.
- Suzuki Y., Tsumoto S., Tanaka H. (1996), Computational Study on Cyclic Structure, using Abstract Rewriting System on Multisets, Proceedings of the International Symposium on Artificial Life and Robotics, January, 38-41.
- Traulsen MI. A., Nowak M.A. (2007), Chromodynamics of cooperation in finite populations, PLos One, March, issue 3, e270
- Takabayashi J., Sato Y., Horikoshi M., Yamaoka R., Yano S., Ohsaki N., Dicke, M. (1998): Biol. Cont., 11, 97-103.

Designing a Passive Folding String Device with an Electromagnet

Atsushi Masumori^{1, 2}, Masayoshi Mitsui² and Hiroya Tanaka²

¹The University of Tokyo, ²Keio University atsmsmr@gmail.com

Abstract

We propose a passively foldable string device. This string device is composed of multiple modules which have an electromagnet on each face. The string is folded according to the pattern of the magnetic poles of the electromagnets, like proteins that fold according to the order of amino acids. It is unfolded if all the magnets have the same polarity. This string can be folded even if the string is long and heavy using an external force, for example a vibration or convection. This kind of device can be used not only as an experimental tool for studying a self-folding process but also as a three-dimensional physical display. In this paper, we propose a design of the folding string device and show some results of experiment for verification of this idea.

Introduction

Self-assembly is a process where some components autonomously organize into structure or pattern without an external direction. This process is ubiquitous in nature, especially in living system. For example, a protein is folded autonomously from a polypeptide according to the arrangement of amino acids constitute the polypeptide.

These kinds of process might be exploited to built an artificial system with the characteristics of scalability, reversibility, self-repairing or adaptability. There are many researches about artificial self-assembly systems from nano scale to *cm* scale [Whitesides and Grzybowski, (2002)] and a few research among them focus on self-folding [Cheung et al, 2011], [Knaian et al. (2012)], [Hawkes et al. (2010)]. Such kind of self-assembly processes are considered to be useful for constructing more complex and smaller object efficiently at nano-scale. Even at *cm* scale, for example, it is useful for manufacturing in an extreme environment such as space or deep-sea, or three-dimensional physical display. In this research, we focus on a self-folding system at *cm* scale.

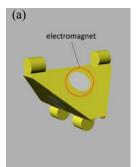
In prior work, Cheung et al. (2011) proposed a string robot which is composed of modular robots and can fold into an arbitrary three-dimensional shape with their method which can generate Hamiltonian path from any given graph and Knaian et al. (2012) developed a small stepper motor and applied it to a string robot. However, these folding robots can be folded by only their motor power, thus the longer and heavier the string becomes, the more difficult a folding is.

We develop a passively foldable string device using electromagnets, which can be folded not only actively but also passively even if the string is long and heavy. In this system, the resulting shape emerges via an interaction of magnetic forces much like amino acids that are attracted and repelled from each other in real proteins. Thus, this device can be used

not only as physical three-dimensional display that can construct three-dimensional shape but also as an experimental tool for studying self-folding process. In this paper, we propose a design of passive folding string device and show its prototype.

Design

In this section, we describe the design of a passively foldable string device. The modules are shaped like right-angled tetrahedron (Figure 1a). A string composed of such a tetrahedron-shaped module can fold into an arbitrary three-dimensional shape and unfolded into a straight line [Griffith. (2004)]. Each module has an electromagnet on each face. These modules are connected by hinge joint (Figure 1b). Thus this string is folded according to the arrangement of the magnetic poles of the electromagnets and if all the magnets have the same polarity, the string is unfolded. For controlling a magnetic pole and a magnetic force of the electromagnets, wireless or wired communication between each module and computer are needed.



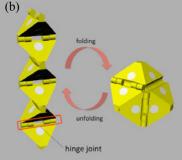


Figure 1: Design of string device. (a) Single module. The module has an electromagnet on each face. (b) String and folded object. The string is composed of multiple modules by hinge joint. The string is folded into and unfolded from three-dimensional object. The string of six modules in the left and an octahedron folded from the string on the right of the figure.

If another permanent magnet is put on each face, the structure, which is folded from a string, can be maintained even if the power supply to electromagnet is cut off (Figure 2). In this case, the force of the electromagnet must be stronger than the force of the permanent magnet.

As described above, the folding robot in prior works use motor actuation for a folding, thus the folding of long and heavy string is difficult. The string device we propose use no motor actuation but electromagnet, and is expected to use external forces in the case of longer string. Therefore, even if the string is long and heavy, a folding can be supported by an external force easily, for example a random vibration to container where the string is or a random convection if the string is underwater.

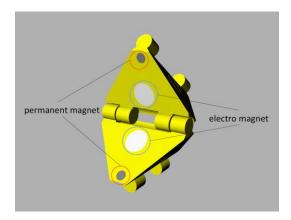


Figure 2: Distribution of electromagnets and permanent magnets on the faces of the modules for maintaining a folded shape.

Implementation

In this section, we show a first prototype of string device. We implemented the prototype for verification of the idea of passively foldable string device. The prototype is composed of three modules, which has no electromagnet but a permanent magnet on their each side (Figure 3). Except for using permanent magnet, the design is the same as described above.





Figure 3: Prototype of string device. Each module was printed by 3D printer (Objet Eden260) and has a permanent magnet on each face. (a) Initial configuration. (b) Folded object.

We conducted a small verification experiment using this prototype composed of three modules. We tested whether the prototype can fold correctly in cases of two kind of magnet; strong (neodym, ϕ 6 × 3mm) and weak (neodym, ϕ 3 × 1.5mm). As the results, in the case of strong magnet, the string could be folded successfully from an initial line-liked configuration without help from an external force. On the other hand, in the case of weak magnet, it was folded successfully only in case of applying an external vibration to a container where the string was (Figure 4).



Figure 4: Passive folding with external force. The string, which is composed of three modules that have weak magnet, cannot fold by itself, but with external force it can be folded.

Discussion

In this paper, we proposed a design for a new type of passive string device and showed its first prototype. The results of verification experiments with a prototype showed that this passively folding mechanism using magnet with an external force could be effective.

This kind of robotic device can be folded into arbitrary three-dimensional shapes and also transformed from one shape to other shapes repeatedly via folding and unfolding process. Therefore, it might be used as a three-dimensional physical display, by which we cannot just watch a three-dimensional object but also touch or use it. It can also be used as an experimental tool for studying self-folding process.

In future work, we implement the string device completely using electromagnets. For this kind of folding string device, the order of folding is important, because if the order is not correct, the string is tangled and fails to fold. Therefore we also need to study the methods for generating an optimized order of folding. We also conduct some experiment using this device, for example, whether the same arrangement of magnetic pattern can be folded into the same structure or different structures.

References

Whitesides, G. M. and Grzybowski, B. (2002). Self-assembly at all scales. Science, 295(5564):2418–2421.

Cheung, K. C., Demaine, E. D., Bachrach, J. R., and Griffith, S. (2011). Programmable assembly with universally foldable strings (moteins). *Robotics, IEEE Transactions on*, 27(4):718–729.

Knaian, A. N., Cheung, K. C., Lobovsky, M. B., Oines, A. J., Schmidt-Neilsen, P., and Gershenfeld, N. A. (2012). The milli-motein: A self-folding chain of programmable matter with a one centimeter module pitch. In *Intelligent Robots and Systems (IROS)*, 2012 IEEE/RSJ International Conference on, pages 1447–1453. IEEE.

Hawkes, E., An, B., Benbernou, N., Tanaka, H., Kim, S., Demaine, E., Rus, D., and Wood, R. (2010). Programmable matter by folding. Proceedings of the National Academy of Sciences, 107(28):12441–12445.

Griffith, S. T. (2004). Growing machines. PhD thesis. Massachusetts Institute of Technology.

Author Index

Ababsa, Tarek, 801

Abdelmotaleb, Ahmed, 965

Ackley, David, 605

Adami, Christoph, 121, 310

Adams, Alyssa M., 522

Agmon, Eran, 514

Alden, Kieran, 614, 622

Alers, Sjriek, 761

Ammar, Haitham Bou, 376

Amos, Martyn, 255

Andersen, Jakob Lykke, 557

Antonioni, Alberto, 368

Arita, Takaya, 105, 239, 415, 778

Armetta, Frédéric, 160

Asakura, Ryo, 539

Aubert, Nathanael, 392

Auerbach, Joshua, 135, 465

Aydin, Deniz, 135

Azumagakito, Tsubasa, 415

Badelt, Stefan, 565

Bailey, Brenae, 847

Bakkum, Douglas, 769

Balch, Tucker, 336

Banda, Peter, 481

Bargar, Robin, 941

Barnes, David, 89

Barreto, Nuno, 588

Beer, Randall, 514

Bentley, Katie, 328

Bernatskiy, Anton, 973

Bersini, Hugues, 360

Bertelle, Cyrille, 849

Biehl, Martin, 949

Bijak, Jakub, 384

Biswas, Ritwik, 641

Bloembergen, Daan, 376

Blount, Drew, 45

Bongard, Josh, 138, 973

Boughman, Janette, 16

Boumaza, Amine, 282

Bourgine, Paul, 532

Bramson, Aaron, 400

Brede, Markus, 856

Bredeche, Nicolas, 158, 272

Brodsky, Micah, 817

Bryson, David, 641

Bryson, Joanna, 47

Buch, Mads, 981

Buckley, William, 819

Buliga, Marius, 490

Bullock, Seth, 263, 368, 408

Burgos, Andres, 352

Burtsev, Mikhail, 457, 771

Butler, James, 614

Cañamero, Lola, 168, 184, 864, 932

Cabri, Giacomo, 192

Cacucciolo, Vito, 230

Calcott, Brett, 701

Canyameres Masip, Sergi, 957

Capodieci, Nicola, 192

Carneiro, Jorge, 734

Caschera, Filippo, 555

Cenydd, Llyr Ap, 473

Charpillet, François, 282

Charrier, Rodolphe, 849

Chebib, Jobran, 736 Cheney, Nicholas, 221

Choi, Insook, 941

Christensen, Anders, 210, 212, 657, 703, 734, 785

Chu, Dominique, 89

Churavy, Valentin, 514

Cianchetti, Matteo, 230

Cieslewski, Titus, 135

Claes, Daniel, 761

Clark, Anthony, 200

Clune, Jeff, 41, 221

Coles, Mark, 614, 622

Correia, Luis, 703

Courgeau, Daniel, 384

Covert, Arthur W., III, 129

Cronin, Leroy, 3

Crowder, Richard, 263

Cully, Antoine, 146, 156

Cussat-Blanc, Sylvain, 541, 801

D'Eleuterio, Gabriele M.T., 681

Daler, Ludovic, 135

Davey, Neil, 965

Davies, Paul C. W., 522, 569

Delord, Evan, 129 Disset, Jean, 541

Djedi, Noureddine, 801

Doursat, Rene, 532

Duarte, Miguel, 210, 657, 703, 785

Duthen, Yves, 541, 801 Dworkin, Ian, 121 Dyer, Fred, 310

Edenhofer, Sarah, 312 Egbert, Matthew, 168 Eiben, A. E., 158 Eita, Mohammad, 631 Elfes, Alberto, 431 Ellefsen, Kai Olav, 649 Erdei, Joachim, 97

Fahmy, Mostafa, 596

Eskridge, Brent, 908

Farahani, Yasaman Majdabadi, 693

Fernández Pérez, Iñaki, 282

Fernando, Chrisantha, 465

Fernando, Pradeep, 135

Ferrante, Eliseo, 299

Flamm, Christoph, 557, 565

Floreano, Dario, 135, 465

Fontana, Alessandro, 447

Francisco, Matthew, 876

Franck, Robert, 384

Freeman, Jesse, 25

Frenoy, Antoine, 23

Fussell, Don, 247

García Sánchez, Pablo, 580

García, Rubén Héctor, 580

Gates, Alexander, 429, 514

Gershenson, Carlos, 427

Gigliotta, Onofrio, 673

Gladden, Matthew, 417

Gold, Jacob, 884

Goldsby, Heather, 121

Goles, Eric, 689

Golestani, Abbas, 693

Gomes, Jorge, 212

Gonzalez, Miguel, 408

Gotoh, Kazuyoshi, 757

Grabowski, Laura, 113

Gras, Robin, 693, 719

Grouchy, Paul, 681

Haasdijk, Evert, 158

Haley, Patrick, 310

Hamann, Heiko, 344

Hanczyc, Martin, 557

Harrington, Kyle, 25, 328, 827, 884

Hart, Emma, 192

Haruna, Taichi, 419

Hassas, Salima, 160

Headleand, Christopher, 473

Hecker, Joshua, 835

Heitz, Gregoire, 135

Hintze, Arend, 365

Hiolle, Antoine, 864, 932

Hofacker, Ivo L., 565

Holekamp, Kay E., 63

Hornby, Gregory S., 973

Howard, David, 431

Hrolenok, Brian, 336

Hubert, Julien, 769

Huepe, Cristián, 299

Hurndall, William, 856

Husbands, Philip, 176

Iba, Hitoshi, 631

Ichinose, Genki, 398

Ikegami, Takashi, 302, 498, 530, 769

Imamura, Yasumasa, 744

Inuzuka, Nobuhiro, 9

Ito, Takashi, 105

Ivanov, Plamen, 274

Joachimczak, Michal, 239

Jocque, Julian, 232

Johnson, Anya, 121

Johnson, Chris, 176

Jung, Jiin, 400

Kato, Shohei, 9, 809

Kauffman, Louis, 490

Kayama, Yoshihiko, 744

Kharma, Nawwaf, 819

Khazanov, Mark, 232

Kim, Daeeun, 750, 759

Kim, Hyunju, 569

Knibbe, Carole, 33

Knowles, Joshua, 79

Koos, Sylvain, 156

Kornatowski, Przemyslaw, 135

Kuruma, Yutetsu, 963

Lakhman, Konstantin, 457, 771

Laschi, Cecilia, 230

Lee, Wonki, 750, 759

Lessin, Dan, 247

Lewis, Matthew, 184, 864, 932

Li. Xun. 439

Lima, Pedro, 734

Lipson, Hod, 41, 221

Logofatu, Doina, 957

Lones, John, 184 Loshchilov, Ilya, 135 Lowell, Jessica, 821, 827

Ma, Xinpei, 629 Macedo, Luís, 588

Maddali, Hanuma Teja, 336 Maesani, Andrea, 135 Magana, Javier, 113 Mangold, Michael, 691 Mariano, Pedro, 212 Marriott, Chris, 736

Maruyama, Norihiro, 769 Mashayekhi, Morteza, 693

Masumori, Atsushi, 539, 769, 987 Matsubayashi, Hideaki, 963 Matthews, Stephen, 47 Mazac, Sébastien, 160 McFetridge, Siena, 129

McGregor, Simon, 498 McKinley, Philip, 148

Mclaughlin, Morgan, 845, 924 Medel, Marco Montalva, 689

Merelo, J.J., 580 Merkle, Daniel, 557

Miikkulainen, Risto, 16, 63, 247, 439

Mirmomeni, Masoud, 365 Mirolli, Marco, 673 Misevic, Dusan, 23 Mita, Takeshi, 769

Mitsui, Masayoshi, 539, 987 Montes de Oca, Marco A., 934

Moore, Jared, 148, 200 Mora, Antonio, 580 Moses, Melanie, 835 Motooka, Daisuke, 757

Mouret, Jean-Baptiste, 41, 55, 146, 156, 455

Muntean, Ioan, 892 Muratov, Sergey, 771 Mutoh, Atsuko, 9, 809

Nakamura, Shota, 757 Naylor, Becky, 290 Nellis, Adam, 506 Nelson, Andrew, 847 Noble, Jason, 408 Noireaux, Vincent, 555 Novitzky, Michael, 336

O'Keefe, Simon, 726 Ofria, Charles, 641

Oliveira, Sancho, 210, 657, 703, 785

Olson, Randal, 310 Onoda, Shota, 809 Parsons, David P., 33 Pavlic, Theodore P., 522 Pentz, Jennifer, 549 Philippides, Andrew, 176 Pickett, Rodney, 121 Pilat, Marcin L., 105 Pitonakova, Lenka, 263 Polani, Daniel, 352, 837, 949 Pollack, Jordan, 25, 821, 827

Power, Daniel, 87 Pugh, Justin K., 202

Rabai, Haifa, 849 Rajagopalan, Padmini, 63 Ranjbar-Sahraei, Bijan, 376, 761 Rasmussen, Steen, 981

Ratcliff, William, 549 Rawal, Aditya, 16 Read, Mark, 290

Regan, Erzsébet Ravasz, 328 Richards, Daniel, 255

Rieffel, John, 232 Rocha, Luis, 429, 876 Rodrigues, Tiago, 210 Roque, Licinio, 588 Rossi, Louis, 934 Ruz, Gonzalo, 689

Sabanovic, Selma, 876 Sakai, Megumi, 983 Salge, Christoph, 837, 949 Sasahara, Kazutoshi, 778 Sayama, Hiroki, 320, 398, 629

Schilstra, Maria, 965 Schlupp, Ingo, 908 Schneider, Eugenia, 691 Scott, Ryan, 719 Shalygo, Yuri, 665 Shell, Dylan, 274

Shell, Dylan, 274 Shenhav, Barak, 815 Shibai, Atsushi, 757 Shoukry, Amin, 631 Silva, Fernando, 210, 703 Silverman, Eric, 384 Small, Trent, 605

Soltoggio, Andrea, 202, 447

Soros, L. B., 793

Stanley, Kenneth O., 202, 793, 900

Stepney, Susan, 71, 506 Stettiner, Orly, 602 Steuber, Volker, 965 Stovold, James, 726 Strauss, Eli, 121

Suzuki, Reiji, 105, 239, 415, 778

Suzuki, Tomohiro, 571 Suzuki, Yasuhiro, 571, 983 Szathmáry, Eors, 87 Szerlip, Paul, 900

Taddei, François, 23 Takahashi, Hirokazu, 769 Takeichi, Yuki, 778 Tanaka, Hiroya, 539, 987 Tanaka, Sayaka, 419 Tarapore, Danesh, 55, 734 Teahan, William, 473 Teuscher, Christof, 481 Timmis, Jon, 290, 614, 622, 726 Timperley, Christopher, 71 Tomassini, Marco, 368 Tonelli, Paul, 455 Travisano, Michael, 549 Tsuru, Saburo, 757 Turgut, Ali Emre, 299 Tuyls, Karl, 376, 761 Tyrrell, Andy, 290

Ueda, Takuya, 963

Veiga Fernandes, Henrique, 614 Virgo, Nathaniel, 498, 530 von Mammen, Sebastian, 312

Wagner, Aaron, 641 Wagy, Mark, 138 Walker, Sara Imari, 522, 569 Wallen, Kim, 336 Wang, Adam, 884 Wang, Olivier, 532 Wang, Yifei, 47 Watson, Richard, 87, 408, 856 Webb, Andrew M., 79 Weiss, Gerhard, 376, 761 Weissman, Daniel, 43 Wenseleers, Tom, 299 Williams, Lance, 711 Wineberg, Mark, 845, 924 Winfield, Alan, 872 Witkowski, Olaf, 302, 392 Wood, Ian, 876 Wrobel, Borys, 97, 447, 965

Yaeger, Larry, 916 Ying, Bei-Wen, 757 Yoder, Jason, 916 Yomo, Tetsuya, 757

Zapata, Octavio, 427